University of Alberta

Fuzzy C-Means and Hidden Markov Models in Modelling Gesture Recognition for Human Machine Interaction

by

Shauna Leanne Rae ©

A thesis submitted to the Faculty of Graduate Studies and Research in
partial fulfilment of the requirements for the degree of Master of Science

Department of Electrical and Computer Engineering

Edmonton, Alberta

Spring 2003

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-82334-2

Canada

University of Alberta

Library Release Form

Name of Author: Shauna Leanne Rae

Title of Thesis: Fuzzy C-Means and Hidden Markov Models in Modelling Gesture Recognition for Human Machine Interaction

Degree: Master of Science

Year this Degree Granted: 2003

Permission is hereby granted to the University of Alberta to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the theses, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatsoever without the author's written permission.

2312 Towne Blvd
Oakville, ON L6H 5J5
Canada

January 29, 2003

Ligeved og naesten
Slaar ingen mand af nesten
Danish Proverb

"Almost or nearly does not
throw a man off his horse."

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommended to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled Fuzzy C-Means and Hidden Markov Models in Modelling Gesture Recognition for Human Machine Interaction submitted by Shauna Leanne Rae in partial fulfilment of the requirements for the degree of Master of Science

Dr. Witold Pedrycz – Supervisor

Dr. Petr Musílek

Dr. Jozef Szymanski

January 28, 2003

To my family and friends,
for providing support,
inspiration, and love.

Abstract

Human Robot Interaction is an important field of research. The easier it is for people to use and co-exist with robots, the more useful robots can be for humans. Gesture recognition plays an important role in Human Robot Interaction. We examine the use of Fuzzy-C means to model posture groups and Hidden Markov Models to model the temporal structure of gestures. Gestures performed by four different people will be captured using simple infrared sensors onboard a Khepera robot. We will propose a behavioural model for the robot response to the human. Recognition rates of 98.1% were achieved for the training data set and 95.9% for the testing set when classifying between "Caress" and "Direct" gestures categories. These rates reduce to 93.3% for the training data set and 90.6% for the testing data set when attempting to discriminate between 8 individual gestures within these categories. Two other categories of gestures were also classified: "Attack", and "Play". The recognition rates between all 4 categories are 90.3% for training data and 89.3% for testing data. These rates reduce to 88.7% for training data and 86.7% for testing data when attempting to discriminate between all 15 individual gestures from the data set.

I would like to acknowledge the following people for their support, ideas, and assistance that made a significant impact on this thesis:

- Dr. Witold Pedrycz (Supervisor)
- Dr. M. G. Chung
- Rob Chapman
- Chet Domanski
- Elliot Christian
- Noah Aklilu
- Tyler Brandon
- Louise Rae
- Marlene Rodriguez
- Francisca Gabarro
- Lino Ramirez
- Leance Baril
- Colleen Mah
- Laurie Rae

Table of Contents

Chapter 1: Introduction 1

1.1 Objectives.. 2

1.2 Behaviour Based Robotics.. 2

1.3 Examples of Interactions... 4

1.4 Gesture Recognition.. 5

1.5 Hidden Markov Models 6

1.6 Clustering techniques 6

1.7 Implementation with Khepera.. 7

  1.7.1 Khepera Specifications.. 7

  1.7.2 A Personable Robot Shell 8

  1.7.3 The Complete Interaction System... 9

  1.7.4 Three Phases of the System.. 10

  1.7.5 The System Model 10

1.8 Overview.. 12

Chapter 2: Human Robot/Machine Interaction... 14

2.1 Defining Human Machine Interaction 14

  2.1.1 Examining the Five Research Areas in HMI 15

  2.1.2 The Many Disciplines of HMI... 17

2.2 Importance of Human Machine Interaction 18

2.3 Human Computer Interaction Past to Present 18

2.4 Human Robot Interaction Development 22

  2.4.1 Towards Programming Robots with Actions and Behaviours 23

2.5 The Future of HMI 25

2.6 Chapter Summary 26

Chapter 3: Gesture Recognition and Data Collection **27**

3.1 Background 27

3.2 Defining Gesture Recognition 28

3.2.1 Descriptive Definition of Gesture Recognition 28

3.2.2 Formal Definition of Gesture Recognition 29

3.3 Examples of Gesture Recognition 31

3.4 Comparing Gesture Recognition to Speech Recognition 32

3.5 IR Sensor Based Gesture Recognition 33

3.6 Gestures Recognised 35

3.7 Variations in Gesticulation 39

3.8 Data Collection of Human Gestures 39

3.8.1 Where to Perform the Gestures 39

3.8.2 When to Start Tracking Gestures 40

3.8.3 Duration of Sampling 40

3.8.4 Sampling Rate 41

3.8.5 Collecting Training and Testing Data 41

3.9 Chapter Summary 41

Chapter 4: Implementation of a Human Robot Interface **43**

4.1 Overview 43

4.2 Hardware Architecture 43

4.3 Data Processing Flow 44

4.3.1 Training Phase 45

4.3.2 Testing Phase 46

4.3.3 Real Time Interaction Phase 47

4.4 Chapter Summary 48

Chapter 5: Pre-Processing 49

5.1 Background 49

5.2 Actual Pre-Processing 49

5.2.1 Two Approaches to Assigning Intervals 52

5.2.1.1 Three Intervals 52

5.2.1.2 Four Intervals 52

5.3 Chapter Summary 53

Chapter 6: Clustering Techniques 54

6.1 Background 54

6.2 Categories of Clustering Algorithms 55

6.2.1 Hierarchical Methods 55

6.2.1.1 Agglomerative Methods 55

6.2.1.2 Divisive Method 56

6.2.2 Graph Theoretic Methods 57

6.2.3 Objective Function Methods 58

6.3 Selecting a Clustering Method 59

6.4 Specific Algorithms of Objective Functional Clustering 60

6.4.1 K-Means Clustering 60

6.4.1.1 The K-Means Algorithm 61

6.4.1.2 Selecting parameters 62
6.4.2 Fuzzy C-Means Clustering 67
6.4.2.1 The FCM Algorithm 68
6.4.2.2 Selecting the FCM Parameters 69
6.4.3 Comparing features of the Objective Function Algorithms 71
6.5 Implementation of a Clustering Algorithm 72
6.6 Results According to Clustering Parameters 73
6.6.1 Convergence of FCM Objective Function 73
6.6.2 Sample Prototypes and Distribution of Data 74
6.7 Chapter Summary 79
Chapter 7: Hidden Markov Models 81
7.1 Finite State Machines 81
7.2 Background 82
7.2.1 HMM Structure 88
7.2.1.1 Ergodic versus Left to Right 88
7.2.1.2 Multivariate versus Discrete Observations 90
7.2.1.3 Number of States 92
7.3 Three Basic Problems in HMM 93
7.3.1 Problem 1 93
7.3.1.1 Forward Procedure 94
7.3.1.2 Backward Procedure 95
7.3.2 Problem 2 96
7.3.3 Problem 3 98

7.3.3.1 Parameter Re-estimation with Discrete Valued Observation Sequences 98

7.4 Implementation of HMM in a Recognition System 103

7.5 Examples of Trained HMM 106

7.6 Chapter Summary 110

Chapter 8: Results of Gesture Recognition 111

8.1 Overall Results 111

8.2 Results According to the Complete Recognition System 111

8.3 Chapter Summary 118

Chapter 9: Conclusions and Future Work 120

9.1 Future Work – The Robot's Response 120

9.1.1 Background 120

9.1.2 Emotional states 121

9.1.3 Motivations 122

9.1.4 Memory 122

9.1.5 State Transitions 123

9.2 Conclusion 125

Bibliography 129

List of Tables

Table 1 – Gestures Recognised by Hèbert 37

Table 2 – Comparing Clustering Methods 60

Table 3 – K-Means Algorithm 62

Table 4 – FCM Algorithm 69

Table 5 – Comparison of Three Objective Function Clustering Algorithms... 71

Table 6 – Summary of Equations for Three Objective Function Methods 71

Table 7 – Implemented Parameters of FCM 72

Table 8 – Sample of Prototypes Resulting from FCM with k=35 75

Table 9 – The Effect of Removing the Zero Instances When Classifying 8 Gestures 77

Table 10 – The Effect of Removing the Zero Instances When Classifying 15 Gestures 78

Table 11 – Parameters of Basic HMM 85

Table 12 – Example of How HMM can Model a Gesture 87

Table 13 – The Forward Procedure 94

Table 14 – The Backward Procedure 96

Table 15 – The Viterbi Algorithm 97

Table 16 – Example of an HMM that Models the Go Left Gesture 107

Table 17 – Example of an HMM that Models the "Go Right" Gesture 108

Table 18 – Example of an HMM that Models the "Hit Right" Gesture 109

Table 19 – Misclassification Matrix for 8 Gestures (Testing Data) 116

Table 20 - Confusion Matrix for the Classification of 15 Gestures (Testing Data) 117

Table 21 – Confusion Matrix for Classification of 4 Major Categories of Gestures 118

Table 22 – Emotional States and Associated Behaviours 122

Table 23 – Emotional State Transitions 124

Table 24 – Summary of Classification Rates 127

List of Figures

Figure 1 – Human Robot Interactions and Corresponding Robot Emotional States 4

Figure 2 – The Khepera Robot 8

Figure 3 – Body Shell for the Khepera Robot 9

Figure 4 – Complete Interaction System 9

Figure 5 – Figurative Representation of Complete Model at Single Time Instance 10

Figure 6 – Research Areas in HMI [4] 15

Figure 7 – Disciplines Contributing to Research in HMI 18

Figure 8 – The Hollerith Tabulating Machine [14] 19

Figure 9 – Wiring the Right Side of ENIAC with a New Program 20

Figure 10 – Electronic Mirror – Interactive Computer Installation [31] 21

Figure 11 – ASIMO Robot Responding to Human Posture [17] 24

Figure 12 – People Beckoning Others to Pose for a Photograph 27

Figure 13 – IR Sensors on the Khepera Robot [24] 30

Figure 14 – Major Components of the Gesture Recognitions System 31

Figure 15 – Measured Reflection Value from IR Sensor vs. Distance to Wall [24] 34

Figure 16 – Measured Reflection Value from IR Sensor vs. Distance to Hand 34

Figure 17 – Hitting the Robot 35

Figure 18 – Tickling the Robot 36

Figure 19 – Capturing or Hugging the Robot 36

Figure 20 – Directing the Robot to the Right 36

Figure 21 – Directing the Robot to Turn Counter Clockwise 37

Figure 22 – Directing the Robot to Stay 37

Figure 23 - Categories of Gestures.. 38

Figure 24 – Hardware Architecture.. 43

Figure 25 – Data Processing Through Training Steps 45

Figure 26 – Data Processing Through Testing Steps....................................... 46

Figure 27 – Data Processing Real-Time Recognition Steps 47

Figure 28 – Distribution of the Occurrence of Grouped Sensor Values Including Extremities 51

Figure 29 – Distribution of the Occurrence of Grouped Sensor Values Excluding Extremities... 51

Figure 30 – Example of Agglomerative Clustering Methods 56

Figure 31 – Example of Divisive Clustering Method...................................... 57

Figure 32 – Comparing Clusters from Graph Theoretic and Objective Function Methods [3] 58

Figure 33 – Examples of Distance Functions from Minkowski Distance Family [11] 65

Figure 34 - FCM Objective Function Converging.. 74

Figure 35 - Distribution Input Vectors with Respect to Prototype 76

Figure 36- Distribution of Input Vectors by Number of Gestures................... 79

Figure 37 – Discrete Markov Process 82

Figure 38 – Ferguson's Urn and Ball Model [36] 84

Figure 39 – An Example of a 4 state Ergodic Model [36] 89

Figure 40 – An Example of a Left to Right Model with the State Jump Limited to 2 [36] 89

Figure 41 – An Example of a 6 State Parallel Left to Right Model [36] 90

Figure 42 – Collection of Models that Represent the HMMs in the Recognition System 104

Figure 43 – Classification Rate According to Number of States and the Number of Prototypes for Training (a) and Testing Data (b) with 8 Gestures Recognized 1132

Figure 44 – Classification Rate According to Number of States and the Number of Prototypes for Training (a) and Testing (b) Data with 15 Gestures Recognized 1154

List of Symbols

x_n	Input of the robot sensor where $0 \le n \le 7$
$\mathbf{x}$	Hand posture at a single time instance
$\mathbf{X}$	Entire data set of all hand postures
$\mathcal{G}$	Gesture, collection of hand postures over time instances
y_j	Cluster prototype of the posture group "j"
$\mathbf{Y}$	Collection of all prototypes
J	Performance index of the clustering algorithm
ε	Stopping threshold for the clustering algorithm
$\mathbf{U}$	Partition matrix for FCM
m	Exponent of the partition matrix
$\mathbf{A}$	Matrix of probabilities of state transitions in HMM
a_{ij}	Probability of state transitions from state S_i to state S_j in HMM
$\mathbf{B}$	Matrix of probabilities of state emissions in HMM
$b_j(k)$	Probabilities of state emissions, of seeing v_k in state S_j in HMM
v_k	Output generated from a HMM at a given time instance
$\mathbf{V}$	Codebook of the HMM containing all posture groups labels, v_k
$\boldsymbol{\lambda}$	used to signify the parameters of an HMM
M	Number of observable output in an HMM
N	Number of states in an HMM
$\mathbf{O}$	Observation Sequence in an HMM
π_i	Probability of starting in state S_i of an HMM

List of Abbreviations

AIBO	Artificial Intelligence Robot (Sony)
AMD	Advanced Micro Devices
ANN	Artificial Neural Network
ASIMO	Advanced Step in Innovative Mobility
ENIAC	Electronic Numerical Integrator and Computer
EM	Expectation Maximisation
FCM	Fuzzy C-Means
FSM	Finite State Machine
HCI	Human Computer Interaction
HMM	Hidden Markov model
HMI	Human Machine Interaction
HRI	Human Robot Interaction
IR	Infrared
LED	Light Emitting Diode
RS-232	Recommended Standard 232 (computer serial interface, IEEE)

Chapter 1: Introduction

Robots are an ever-increasing presence in the developed world. They are used extensively in manufacturing facilities to assemble cars, cut fabric, sort, and build electronic circuit boards. There are service robots with advanced autonomy found in hospitals as delivery agents, museums as tour guides, in police squads to search out bombs, and in the field where they work as pack mules or surveyors. Robotic devices are being developed to help the disabled in many functions. There are even sophisticated robots found in the home in the form of pets or toys.

Many of these robots could be more useful or entertaining if they contained enriched Human Robot Interaction (HRI) in their systems [26]. HRI enables people to more easily take advantage of what robots are capable of doing and work with them towards an end goal. In tasks where human strength is insufficient, a robotic strength can assist the human and the knowledge of the person can assist the robot [34, 47]. HRI also makes it possible to work in dangerous situations, such as recovery operations or exploration. It also plays an important role in studies relating to cognitive psychology [5, 6, 8, and 37]. When concerned with toys or pets such as Sony's AIBO [39], human interaction with the robot makes them considerably more dynamic.

There are a few areas where research into this problem is continually required [1]:

> "It is expected that robots of the future will coexist and interact with humans in places such as homes and offices. In order to make the interaction of service robots with people safe and effective, new technologies must be developed, ranging from physical material design for robot bodies, to intelligent controllers."

1.1 Objectives

The focus of this research is on the developing a component of an intelligent controller, more specifically gesture recognition. We will explore how human interactions can affect the "emotional" state and subsequently control the behaviours of a robot. While interaction between human and robot can take on many forms, this work will concentrate on human gestures recognised through the aid of very basic sensors.

Our goal is to build a gesture recognition system using basic IR sensors and considering natural, intuitive movements. We begin by extracting hand postures and positions and assigning them to posture groups via clustering. We will model the temporal changes of the posture groups with Hidden Markov Models. The trained recognition system will identify gestures performed by a person.

A further goal is to explore the concepts of behaviour-based robotics [2] in the architecture of an HRI system. The sequence of the gestures could determine the behavioural state of the robot, which would stimulate further actions from the person. Movement and/or light indicators would exhibit the resulting robotic behaviours. We implement the system on a Khepera robot interfaced over an RS-232 serial line to a Linux computer. The name of the robot is Hèbert.

1.2 Behaviour Based Robotics

Traditionally, robots have been programmed to move in a known environment. The programming is based primarily on the exact position of where the robot, or robot arm, needed to be. For example, if we want a robot of pick up a microchip and place it in a circuit board. We would tell the robot exactly where to pick up the chip and exactly where to place it on the board. The approach is top down in nature and inflexible to changes in the environment; say for example the board was in backwards the robot would not be able to complete its function unless it was programmed to flip the board.

An alternative approach to robotics is a behaviour-based method. Instead of having a complete representation of all aspects of the environment and corresponding actions programmed into the machine, the robot is trained to react to its environment. This method, sometimes referred to as reactionary, allows the responses, or behaviours, of the robot to emerge from its surroundings. The main advantages of this approach are that machines have real-time responses and require little representation to perform the same tasks as deliberative machines.

Some of the original work in behaviour-based robotics was done long before the term was coined. Norbert Wiener is credited with the early development of cybernetics, the marriage of control theory, information science, and biology, in the late 1940s [2]. His work with R. W. Ashby, another cybernetics pioneer, expressed natural behaviour in terms of the mathematics used for feedback control systems. In 1953 W. Grey Walter applied this principle to develop Grey Walter's tortoise, a robot which was able exhibit behaviours of seeking light, heading towards weak light, backing away from bright light, avoiding obstacles by turning and pushing, and recharging a weak battery [45]. The tortoise guided itself using a light sensor connected to the steering mechanism of a single driving wheel.

Most behaviour-based robots have no memory and are purely reactive, but some systems incorporate the history of past environmental stimulus into a more responsive system [2].

The work in this thesis will examine interactions in terms of behavioural responses to gestures, which are temporal in nature. The history of gestures will affect the "emotional" state of the robot. The current state and gesture will determine the robotic behaviour.

1.3 Examples of Interactions

Some examples of these interactions are listed below. First, consider some of behaviours that the human might exhibit. (NOTE: the numbers correspond to those in Figure 1.)

1. Hug or capture the robot (surround robot with hands)
2. Tickle the robot (quick short movements with one finger)
3. Hit the robot
4. Give the robot a direction command
5. Time elapsed with no recognized gesture

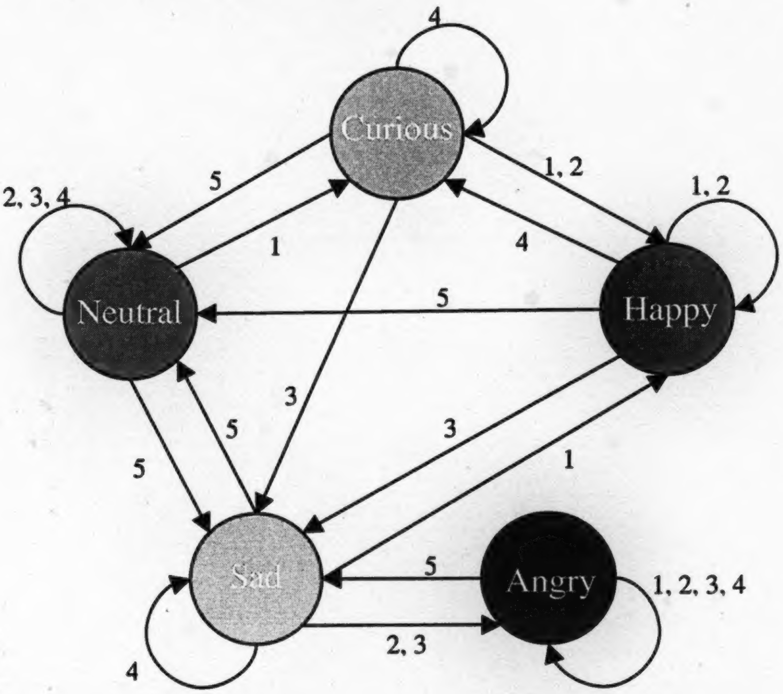


Figure 1 – Human Robot Interactions and Corresponding Robot Emotional States

Figure 1 shows a simplified version of the interactions to give a concept of how the interactions could occur. It does not factor in all accounts of time.

Repeating gestures in a given state would actually cause different transitions than those represented in Figure 1.

Suppose that Hèbert is exploring an environment and then a person approaches it. The person greets the robot by hugging it. As with an animal, this will affect the next state of the robot and it will become curious. If subsequent gestures are aggressive, such as hitting, the robot will enter a state of sadness. If, instead, the person played with the robot, by tickling, it would become happy instead of sad. A number of different interactions are possible.

1.4 Gesture Recognition

While there are a number of research areas regarding HRI, the primary concern of this thesis is to examine gesture recognition. We define a gesture as movements of the human body meant to communicate a thought or intention. They are not limited to movements of hands and arms, but also include facial expressions and body language in general.

In some studies, information is conveyed to the robot using explicit signals generated by flashlights [41]. This requires advanced cameras and unnatural movements and props on the part of the human. Our system allows the user to move his or her hands in free space in intuitive ways.

Our interest is to determine a way to model each gesture so that we can represent it with a computer. The models include both spatial and temporal information of each gesture. Once accurate models exist for each gesture, they can be used in a classification system. There are a variety of algorithms that can model gestures. They include template matching, artificial neural networks (ANN), as well as Hidden Markov Models (HMM).

1.5 Hidden Markov Models

The observable output of a gesture, like the output of many other real world processes, can be considered a signal. We need to find a way of representing each gesture by some model. This will allow us to use the models in a classification system. As such we can build signal models of each gesture.

There are two main types of signal models: deterministic and probabilistic. Deterministic models are useful when a signal can be broken down into known components, such as sine waves in communications systems. Probabilistic models are useful when a signal is more easily represented by a collection of statistical parameters. Hidden Markov Models (HMM) are probabilistic signal models because they represent the signals in terms of probabilistic state transitions and state emissions [36].

Probabilistic models can more easily represent gestures (such as movements of a person around a robot, physical contact with a robot, and spoken words) than deterministic models. Gestures contain temporal information and the states, and transitions of HMMs represent this well. In the past, HMMs have been successfully applied towards the problem of gesture recognition [40, 48-49]. Given proper set-up of the problem, success rates of greater than 93% of correct classifications have been obtained on testing data sets.

1.6 Clustering techniques

Data used to generate hidden Markov models can be either multi-dimensional or discrete single-dimensional. Although using continuous data may offer better classification performance, it is computationally more expensive. As such, data is often clustered and classified prior to implementing the hidden Markov models.

Clustering is the process of finding structure in data. K-means clustering is often partnered with HMM. It is a simple clustering algorithm, which extracts cluster prototypes from the data and assigns the data points to one of the clusters based on a distance function.

Fuzzy C-means (FCM) can also be used to determine the clusters. It is similar to the K-means algorithm but allows for fuzzy boundaries between clusters. Each data point has a degree of membership in each of the clusters instead of belonging completely to only one of the clusters.

In the case of gesture recognition, clustering is often used to find grouping of similar hand or body positions or postures.

1.7 Implementation with Khepera

In order to implement a system that allows for human robot interaction a Khepera [24] robot will be used.

1.7.1 Khepera Specifications

The basic Khepera robot is a small cylindrical robot, that has a 5 cm diameter and a 5 cm height, see Figure 2. It has a 16 MHz Motorola MC68331, with 256kb of RAM, and a serial port with data rates of up to 38400 Baud. It is equipped with 8 infrared (IR) proximity and ambient light sensors that are located around the perimeter of the robot. There are 6 located in the front and 2 at the rear. The sensors have nearly 5 cm IR range.

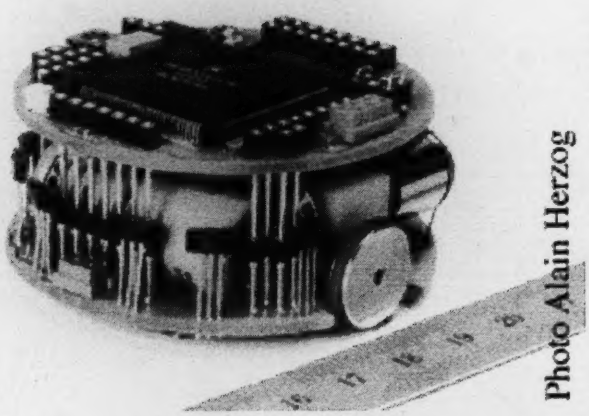


Photo Alain Herzog

Figure 2 – The Khepera Robot

It is also possible to add a number of extension turrets to the Khepera. These include more sophisticated vision sensors, a gripper module, a general purpose input/output, I/O, board, and a radio transmitter.

One of the goals of this work is to achieve human robot interaction with only simple sensors. As such, only the IR sensors will be used in this project.

Since the Khepera has limited on board memory and a relatively slow processor, much of the data processing will take place on a host computer. This computer will be connected to the Khepera by the RS232 serial port.

1.7.2 A Personable Robot Shell

Chet Domanski designed a body shell with the goal of encouraging interaction with the robot. The shell has openings around all of the IR sensors. Two LEDs currently onboard the Khepera would be placed in the eye sockets on the body shell. A slot at the top allows the user to connect a serial tether to the host computer. Figure 3 shows a picture of the body.

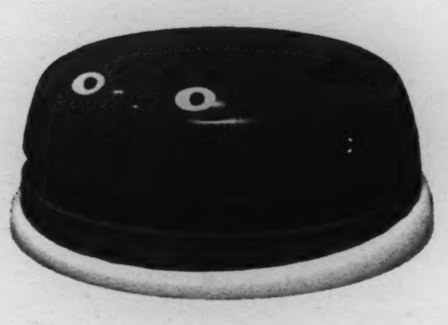


Figure 3 – Body Shell for the Khepera Robot

1.7.3 The Complete Interaction System

The entire interaction systems consist of a human, the Khepera, and a Linux based Pentium computer. The robot detects hand movements with the IR sensors, captures the data with the microprocessor, and then sends it along the RS232 serial connection to the host computer. The host computer processes the data and performs gesture recognition and would contain the behaviour model. Figure 4 gives a representation of the system in the real world.

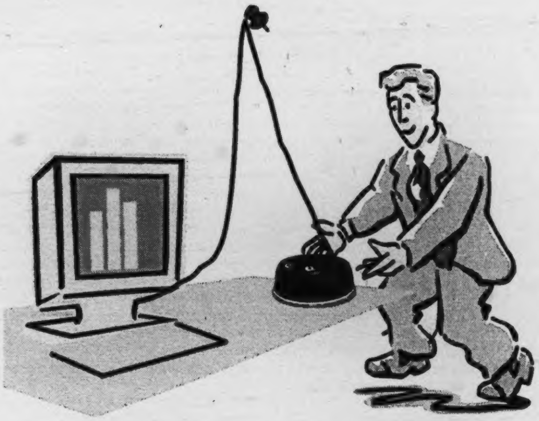


Figure 4 – Complete Interaction System

1.7.4 Three Phases of the System

We will identify three main phases of this experiment. There is the training, the testing, and the real time interaction phase. The training phase is the phase during which the models are trained and developed. The testing phase occurs when the models are verified for accuracy. The real time interaction phase occurs once proper models have been designed and built; it involves real time gesture recognition and robot behaviours.

1.7.5 The System Model

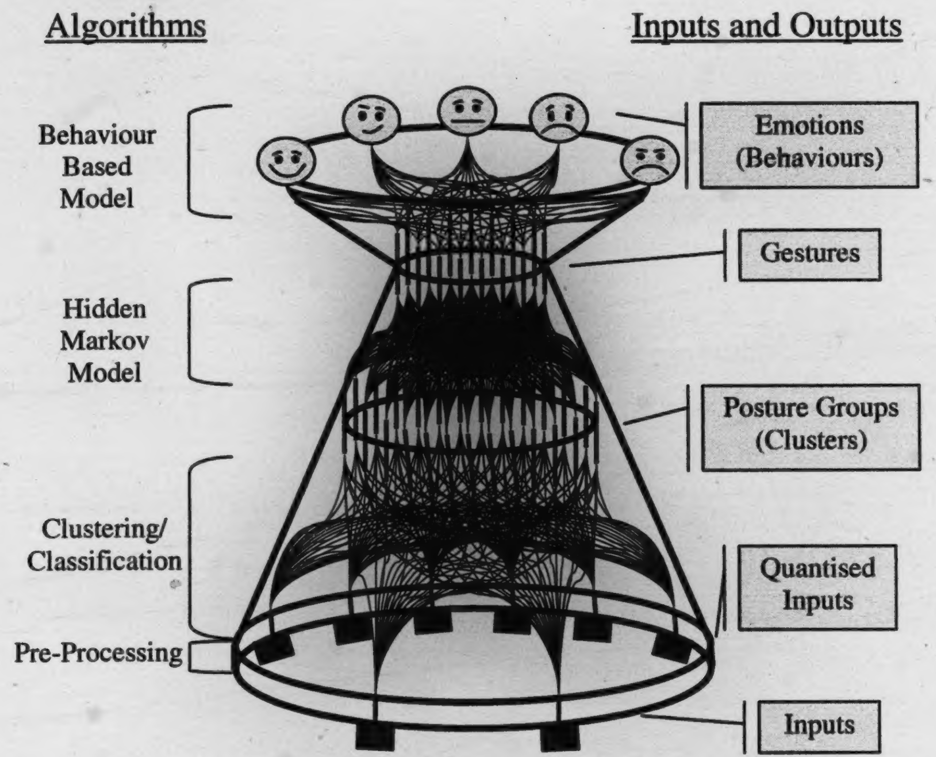

Algorithms
Inputs and Outputs
Behaviour Based Model
Hidden Markov Model
Clustering/ Classification
Pre-Processing
Emotions (Behaviours)
Gestures
Posture Groups (Clusters)
Quantised Inputs
Inputs

Figure 5 – Figurative Representation of Complete Model at Single Time

Instance Figure 5 gives a basic representation of the complete robotic system. It shows the path from the input of the sensors to the output of different behaviours. The left hand side of the diagram shows the different algorithms required to process the data acquired. The right hand side displays what is input and output from each stage in the system.

We will build the model from the bottom up. The first stage is pre-processing. This stage is responsible for converting the raw sensor data into data that is more suitable for the clustering process.

The next step is the clustering stage. During the training phase of the experiment, the clustering stage receives the data from pre-processing and finds posture groups within the data. Essentially finding out which hand positions can be represented by the same group. The prototypes of these groups are used to quantize each instance of data by labelling it with a cluster group. During testing and real time implementation phases of the experiment, the only action is to assign the data to one of the clusters.

The HMM stage follows. The resulting quantised data is passed on to the HMM as a cluster label. During the training phase, a sample of sequences of each gesture is used to train a model to represent each individual gesture. During the testing phase, gesture sequences are matched to the models. Once correct models are trained, they are used in the interaction phase. In this phase incoming sequences are once again matched to a model. The output of this stage is a gesture label.

Gesture labels are passed onto the behaviour based model. While it is possible to train a behaviour-based model, that is beyond the scope of this experiment. As such, there is no behaviour-based model involved in the training and testing phases. We propose a hard coded behaviour based model as future work. Once implemented, gesture labels serve as inputs to the model during the interaction stage. The emotional state of the robot would change as different gestures occur.

1.8 Overview

Chapter 2 presents an overview and definition of Human Machine Interaction. It reviews the history of both human computer and human robot interaction. It highlights state of the art developments in this area and looks at what the future may bring.

Chapter 3 provides both a descriptive and mathematical definition of gestures and gesture recognition. It presents a literature review of research into gesture recognition. We, then, detail the gestures recognized by the system and conclude by explaining how gestures were collected for training and testing.

Chapter 4 presents the hardware architecture and the data processing flow of the system for all phases of the system.

Chapter 5 presents the pre-processing stage of the system. We provide details concerning what pre-processing attempts to accomplish and show how it was actually implemented.

Chapter 6 presents clustering algorithms. It gives a definition of what clustering does and an overview of the three major categories of clustering methods. It then gives guidelines on how to select a clustering method given a particular problem. It continues by detailing objective functional clustering, namely K-Means and Fuzzy-C Means algorithms and concludes by explaining how the Fuzzy-C Means algorithm was used to classify posture groups.

Chapter 7 presents Hidden Markov Models. It provides background information on what Hidden Markov Models are and how they apply to gesture recognition. It explains the solutions to the three basic problems in Hidden Markov Models and shows how they are implemented in this recognition system.

Chapter 8 presents the results of the gesture recognition system. It highlights the classification rates and analyses the outcome of the system.

Chapter 9 gives concluding comments on the experiment and future goals of this research. The future work includes a behaviour model for the interaction system. It shows how the robot responds to gestures by exhibiting different behaviours in different emotional states.

Chapter 2: Human Robot/Machine Interaction

This chapter will present a definition and background information for Human Machine Interaction (HMI). It will discuss the importance of research in HMI. It will highlight the developments of human computer interaction and human robot interaction and show how the progress in these areas leads to increased ease of use and wide spread implementation. Finally, it will present an overview of the future of HMI.

2.1 Defining Human Machine Interaction

The study and implementation of human robot interaction, HRI, is a subsection of the broader field of HMI. HMI is also called human computer interaction, HCI, man machine interaction, MMI, or computer human interaction, CHI. Simply defined, "*HMI is the study and theory of the interaction between humans and complex technology, generally computers* [4]."

The actual study of HMI occurs on a number of different levels and from different perspectives. Booth further explains the meaning of HMI by breaking it down into the five different research areas shown in Figure 6. They are:

- research into organisational impact,
- research into design,
- research at the task level,
- research into matching models, and
- research into interactional hardware and software.

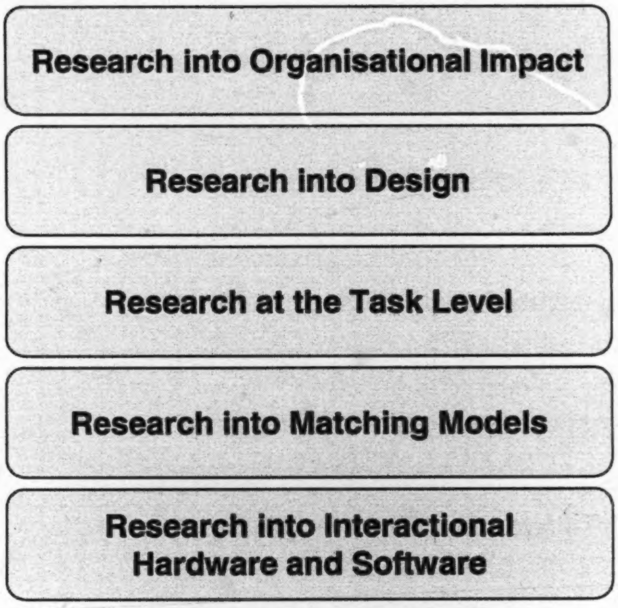

Research into Organisational Impact
Research into Design
Research at the Task Level
Research into Matching Models
Research into Interactional Hardware and Software

Figure 6 – Research Areas in HMI [4]

2.1.1 Examining the Five Research Areas in HMI

At the organisational impact level, studies examine how computers and machines affect companies, groups and organisations. Researchers look at how the duties and responsibilities of individuals change and aim to minimise difficulties, job deskilling, and conflicts between groups with the introduction of new technologies. Take for example the introduction of industrial robots into a manufacturing facility. This area would study the impact of their implementation.

Design and development in HMI has two main focal points. They include examining the individuals involved in the design process as well as the design and development itself. Ultimately, the goal is to design systems which are more user centred rather than simply technology centred. Take, for example, modern programmable thermostats. They provide users with enhanced functionality by allowing a person to select different settings at different times a day. They allow people to customise the temperature settings for various activities, such as sleeping, being awake at home, or being at work. The benefits include improved comfort and energy savings. Unfortunately, many users get very confused if they attempt to program the device and they end up being underused. The interface to a traditional thermostat is easy, simply turn the dial until the arrow points to the temperature we want and it is set. A well designed system must consider the end user to be an important, if not central, factor.

Research at the task level involves developing methods that ensure that systems fulfil user needs. Designers of systems need to know the end users functional and information needs and understand that these needs may vary from user to user. A robot vacuum cleaner would be useless unless it made sure to remove all the dirt of the floor and not just go over every spot once. The systems must also allow the users to perform tasks in manner that they would like. This is an area where machines that can adapt to a user would be beneficial so that users would not have to adapt the machines. An autonomous vacuum cleaner should not start vacuuming in the middle of the night, unless its owners worked night shifts. The frequency that the vacuum cleans would also depend on the user; a home with family with small children would likely require more frequent cleaning than the home of a single person.

The forth research area in HMI is the study of matching models. Researchers here try to determine the best match between the computer or system model of a task and that of the user. They also work towards creating better, more useful models of user knowledge and incorporating them in such a way that they help the user complete their tasks. Consider, for example a medical diagnosis system, where doctors can enter the symptoms of the patient. Even though the doctor has knowledge of diseases and their associated symptoms, such a system would hope to increase reliability and reduce human error. A number of other expert systems are being developed including work at Daimler Chrysler, which will automatically design custom semi-trucks according to customer specifications. This model incorporates knowledge of the engineers and the designers and aims to reduce the number of person-hours per vehicle. [23]

Finally, there is the research into interactional hardware and software. This area is the focus of this thesis. Researchers here are concerned with how current input and output technologies affect interactions and with the consequences of new techniques. These new technologies include improvements of speech recognition and generation as well as gesture recognition and generation. The goal is not just to develop new interactional technologies but also to determine where and when these new techniques should be applied. In the case of this research, concern is for how to best recognise gestures in a systems with simple IR sensors and how the robot should respond to them. Gesture based interaction is chosen because it is part of the natural interaction between humans and humans and between humans and animals. It seems that humans would find gestures a natural way to interface with a robot especially when directing it.

2.1.2 The Many Disciplines of HMI

Just as there are a number of different research areas in HMI, there are a number of different disciplines that contribute to the literature and advancements in the field. They include: ergonomics, software engineering, mathematics, cognitive science, organisational psychology, social psychology, sociology, cognitive psychology, computational linguistics, and artificial intelligence. Researchers in each of these disciplines contribute to one or more areas of study such as what is shown in Figure 7.

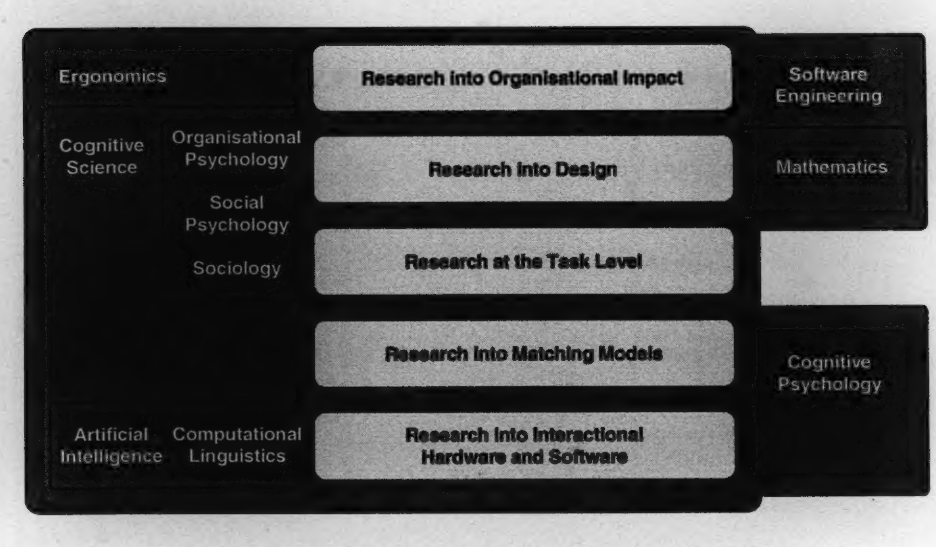

Ergonomics
Research into Organisational Impact
Software Engineering
Cognitive Science
Organisational Psychology
Research into Design
Mathematics
Social Psychology
Sociology
Research at the Task Level
Research into Matching Models
Cognitive Psychology
Artificial Intelligence
Computational Linguistics
Research into Interactional Hardware and Software

Figure 7 – Disciplines Contributing to Research in HMI

2.2 Importance of Human Machine Interaction

As with all research, one must ask, why study this topic? Why is it important? A simple reason is that the failures of a number of products that were thought to be useful and beneficial are attributed to a poor user interface; this makes sense. If people cannot understand how to use a machine or if they are physically incapable of using it, they will not use it.

2.3 Human Computer Interaction Past to Present

Since the dawn of computers, our ability to interface and interact with them has become increasingly more natural and intuitive.

Computer development began even before the turn of the 20th century. Mechanical switches were used long before phototubes or electrons. The machines created were able to complete calculations much faster than humanly possible, but their interfaces were not very instinctive.

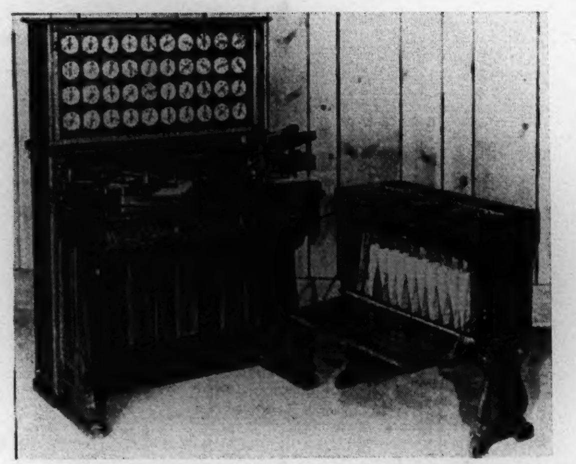


Figure 8 – The Hollerith Tabulating Machine [14]

Developed in 1890, one such machine was the Hollerith tabulating machine, see Figure 8, that was used to help take the US census. Originally, there was concern that the job of enumerating all the residents of the country would take 10 years. The Hollerith was able to complete the task in less than a year. Data was input into the machine by using punch cards. One could wire up pins and it would count, for instance, how many farmers in Idaho were of Irish descent, lived on less than 20 acres of land, had at least 3 children and made more than $50 a year. Dials on the front of the machine displayed the results [44].

One of the first electrical computers used extensively in the US was the ENIAC, 1946. The vacuum tube based technology was used primarily for ballistic projectile calculations in World War II. The interface to the device was, most certainly, not intuitive. In order to enter a new program into the machine, the device had to be rewired, see Figure 9. This was a very tedious and error prone process. The machine also had a bank of dial switches that were used as look up tables. Von Neumann suggested using the dial switches to program the computer instead. The wires were then set up with a program that referenced the dials for programming input. Even with this major improvement in interfacing, only experts could use the computer.

Figure 9 – Wiring the Right Side of ENIAC with a New Program

Standing: Ester Gerston, Crouching: Gloria Gorden (?) U.S. Army Photo (from the archives of the ARL Technical Library)

It took a vast number of improvements to input devices before the average person could use computers. From punch cards through keyboards and onto the introduction of a computer mouse, the methods people used to input information into computers changed dramatically. With each improvement more and more people we able to use the technology. Input devices for personal computers have evolved to include microphones, cameras, drawing tablets, joysticks, steering wheels, and rudder, gas, and brake pedals.

The development of output devices and methodologies has been equally as important for bringing computer technology to the masses. A monitor capable of displaying text and images was able to close the interaction loop far better than banks of indicator lights and printouts. The now popular menu driven, windowed style of operating system was developed at Xerox Parc and later became the inspiration for the mass-produced Apple Macintosh [32].

Human computer interaction has improved dramatically. Even my 87-year-old grandmother, who had never considered using a computer before she turned 84, can play bridge online and chat with friends and family using messaging services. This being said, she encounters a number of difficulties when attempting everyday tasks, such composing an email and cannot always understand why the computer is "acting up" for various reasons. Many of these difficulties can be attributed to things such as not knowing when to hover a mouse cursor, click, or double click or which window is currently active. We are still a long way from a natural interface.

HMI is not limited to computers that sit on a desk or giant old machines the size of city blocks. Today microcomputers are found everywhere; remote controls, microwaves ovens, elevators, vending machines, and telephones are just a few examples of devices that contain computers.

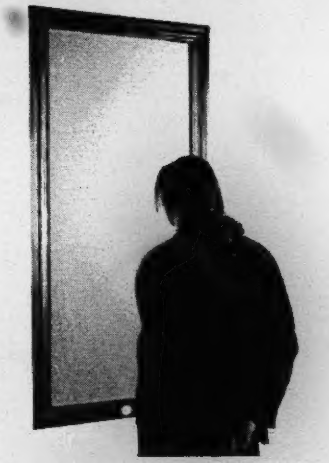


Figure 10 – Electronic Mirror – Interactive Computer Installation [31]

Many children of today are presented with many highly interactive toys that contain computers and will become familiar with interacting with machines before they even know how to read. Consider some of the toys produced by Leapfrog such as the LeapPad® Learning System can read along with children and has interactive learning games. Perhaps this upbringing will alter the definition of what natural interaction is. Certainly most children already find current computer interfaces more natural than the average 60 year old does.

The best interface is one where the user does not even realise they are interacting with a machine. Consider an electronic mirror developed by at UCLA. The mirror is built with a liquid crystal layer on top of an actual mirror. Sensors detect how close a person is to the display and a computer then alters the transparency of the liquid crystal layer. As the distance changes the image becomes more or less crisp and visible [31]. This is an example of a simple interface that is transparent to the user. Will interacting with a robot become equally as transparent?

2.4 Human Robot Interaction Development

A device that is considered one of the precursors to robotics is the Grey Walter Tortoise [2, 45]. Its actions were determined by the circuit that was designed and built by inventor W. Grey Walter. The actions could not change once the circuit was built and offered no real human interaction.

Like the Grey Walter tortoise, a number of mechanical devices, built since the beginning of the industrial revolution, have served only one role. Devices built to perform a specific task, such as the production of manufactured goods, could not be programmed. Thus, once the product they produced became obsolete, they did too.

One product of this era was the Jacquard loom. It was one of the first machines to offer a large degree of freedom. Punch cards were used to input which pattern to weave.

It was clear that mechanical devices to assist in factories would be more cost effective and useful if they could be adapted to perform different tasks. In

1954, George Devol filed a patent for a programmable robot arm. Just like the Jacquard loom, the movements were controlled by punch cards [32].

2.4.1 Towards Programming Robots with Actions and Behaviours

By 1958 Devol and Joe Engleberger founded Unimation and built the first industrial robots. Shortly there after, they developed robots that used magnetic memory instead of punch cards. The robot movements could be programmed by simply moving the robot arm along the desired trajectory of motion. The movement pattern was stored in memory and could be repeated.

As computers became more affordable robotics research grew, and soon robots in manufacturing were equipped with sensors and even cameras. These types of input devices enabled robots to interact with their environment and humans. By 1986 Naval Ocean Systems developed a robot proxy that captured the motions of a person and duplicated them by a robot. The robot was also equipped with two cameras for eyes. These images were projected back to the operator who was wearing bulky glasses with video monitors attached.

Currently there is a great deal of research devoted to human robot interaction. There are robots taking on human form, such as COG, Kismet, or ASIMO, or pet form, such as AIBO.

COG was built as the COG shop, now the humanoid research group, at MIT. COG is a humanoid robot without legs. The project has many goals, including developing human level intelligence. It is believed that the only way to achieve such intelligence is if the robot can interact with humans more naturally and thus requires a human form [8].

Kismet is a robot that is based on COG and is constructed as a humanoid head. Kismet has additional facial features, including articulated eyes, ears, and mouth and is used primarily to model an infant-caregiver relationship. The human is the caregiver and the robot is the infant. The robot displays its emotional state by changing its facial features. The human interacts with Kismet by moving his or her face near the robot or stimulating the robot with different toys. The robots "emotional" state changes according to the level and type of stimulation [5, 6].

AIBO from Sony is a pet robot designed to behave in many ways like a dog. The AIBO can run different programs including one that allows the user to "raise" AIBO through maturing stages from baby through to adult. The way we interact with the robot affects its behaviour and can cause it to become more sociable or more independent as it matures. It has 20 actuators, a number of indicator lights, and a speaker as well as touch sensors at specific locations, a camera, and microphone. It uses eye lights, ears, and tail movement to display behaviours and entertain.

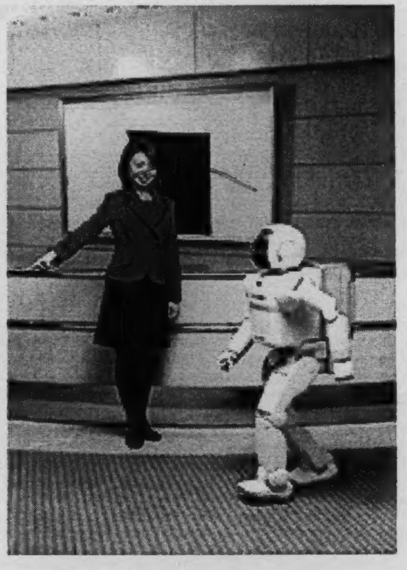


Figure 11 – ASIMO Robot Responding to Human Posture [17]

Since 1986 Honda has been working on a humanoid robot. The robot is called ASIMO, which stands for Advanced Step in Innovative Mobility. The Robot is capable of walking and climbing stairs. "Honda hopes that the time will come when humanoid robots play an important role in serving us and enriching our lives and society [17]."

In December of 2002 Honda announced a new intelligence model for ASIMO. The robot is now capable of recognizing faces, and sounds, as well as a limited number of body postures and hand gestures. It can greet people, address recognized people by name, and relay messages to them.

2.5 The Future of HMI

Moravec envisions a day when the interaction between human and computer or robot is as seamless as the electronic mirror in the example above. He describes a system where the user wears what he calls magic glasses and wears a special coat. The glasses project a scene directly to the user's eyes and the coat becomes a direct connection to the person's sense of touch. A robot in a completely different environment could project the view seen by its "eyes" to the glasses and transmit the "feelings" felt to the coat. Ultimately, the user would not sense the glasses or the coat but only the experiences of the robot.

He also proposes a system that allows the human mind to be transferred into a computer one layer at a time, thus allowing the mind to be copied to multiple computer systems and a mind to "live" eternally.

Service robots will likely become more a part of our daily lives. They will become increasingly helpful in medicine, in both surgery and prosthetics, in the workplace, and with household chores. Engleberger, who was instrumental in the development of the first industrial robots, is now working to develop robots that can "care" for the elderly, allowing them to remain out of nursing homes for a longer period of time. However, can we ever create a robot that gives a person a true sense of being cared for?

2.6 Chapter Summary

Humans have used machines more and more extensively as time has progressed. These machines have enable people to complete jobs faster and avoid having to repeat tedious computations and procedures. A number of different disciplines have and continue to contribute to improvements in the field of HMI.

We have shown that as machines become easier and more intuitive to use, their use becomes more widespread and more people are able to benefit from their applications.

We must be sure that the development of human interactive robots will be beneficial in all ways and contributes to society as a whole. We must determine where they can appropriately replace the duties of a human and where they are infringing upon humanity.

Chapter 3: Gesture Recognition and Data Collection

This chapter will present a background of what gestures are. It will provide both a descriptive and mathematical definition of gestures and gesture recognition. We, then, provide exampl[illegible] gesture recognition in the literature. We will then examine how gesture recognition compares to speech recognition.

We will then turn our focus to the application of gesture recognition to this experiment, examine infrared sensor based gesture recognition, and present the actual gestures recognized. We will finish the chapter by discussing how the gestures were actually collected and how we numerically represent the data.

3.1 Background

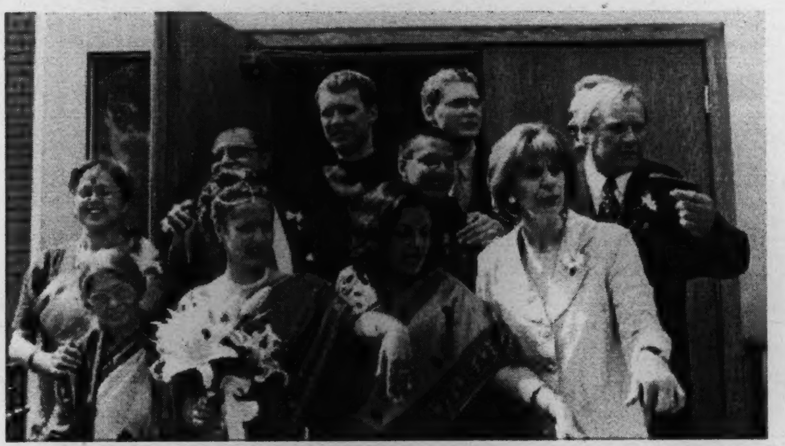


Figure 12 – People Beckoning Others to Pose for a Photograph

Gesture recognition is an important subset of research into HMI. People use gestures naturally and intuitively on a regular basis in their daily lives, often unconsciously. Figure 12 depicts people from different cultural backgrounds beckoning people to pose in a photograph. In [19], Huang describes humans as experts at using gestures to communicate. It then seems sensible to exploit this natural "expertise" when trying to improve the interface between humans and machines. Consider how a mouse or keyboard significantly hinders the natural flow of computer-based presentations to an audience. Have you ever seen an able-bodied person go to get their mouse or keyboard to try to communicate with able-bodied people face to face?

3.2 Defining Gesture Recognition

3.2.1 Descriptive Definition of Gesture Recognition

Let us begin by defining what a gesture is. According to the Canadian Oxford Dictionary [9], a gesture is defined as:

1. Movement of a limb or the body as an expression of thought or feeling.
2. The use of such movement especially to convey feeling or as a rhetorical device
3. An action to evoke a response or convey intention, usually friendly.

In [27], Liang defines gestures as, "Hand and body movements that can pass information from one to another." [18] hierarchically breaks down different categories of gestures.

Intentional gestures are ones performed deliberately, whereas unintentional ones are performed unconsciously, such a blinking or sometimes smiling, frowning, or tapping ones fingers. [18] further breaks the intentional category into verbal and non-verbal groups. The verbal group encompasses sign language whereas non-verbal gestures do not directly represent words.

Machine based gesture recognition is the process by which gestures are captured quantitatively, by some method, and then compared to generated models of desired gestures. The gesture is identified by the model that best represents it.

3.2.2 Formal Definition of Gesture Recognition

More formally, consider gestures, $\mathcal{G}$, as postural changes versus time [27]. In the case of a recognition system they are represented as changes in body posture and position, $\mathbf{x}$, over discrete time moments, $k=1,2,\ldots T$, where T is the total number of samples taken of a particular gesture. For simplicity, we will assume that posture also includes position for the remainder of this discussion.

Many gesture recognition methods break the problem down into three stages. The first stage involves determining how to quantify postures. The second stage involves spatial recognition, recognising the body postures. The third stage involves temporal recognition, recognising how the postures change over time.

The first challenge in gesture recognition consists of trying to determine how to model the posture, $\mathbf{x}$ at any instance in time. With data gloves, this can take the form of the actual sensor readings or the angle of bend in each joint of the hand and the position of the hand. In vision based gesture recognition, the selected body part must be extracted from the image and subsequently features of the image need to be tracked.

In the case of this experiment the posture can be represented in terms of the vector of the IR sensor values, $\mathbf{x} = \{x_0, x_1, \ldots, x_7\}$. A posture at a specific instance of time is denoted as, $\mathbf{x}(k) = \{x_0(k), x_1(k), \ldots, x_7(k)\}$. Finally a gesture is defined by, $\mathcal{G} = \{\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(T)\}$. Figure 13 shows how the sensors are placed around the robot and how they relate to hand posture. The range of the sensors for detected a hand is 4 cm.

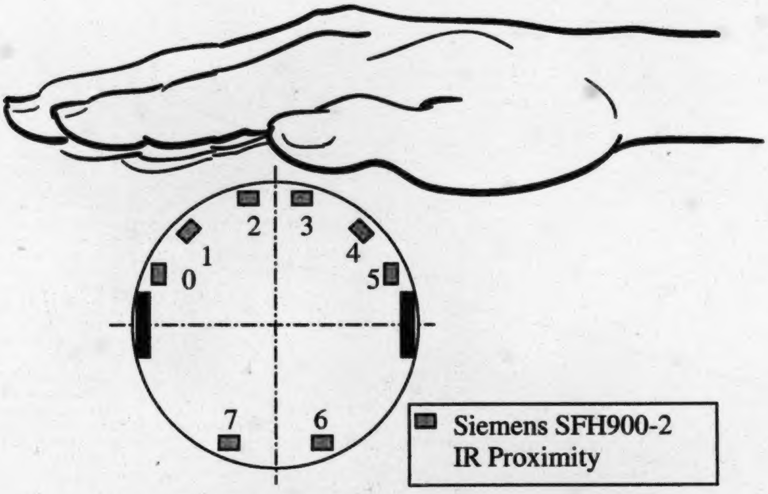


Figure 13 – IR Sensors on the Khepera Robot [24]

The raw sensor data are pre-processed; each individual sensor input, x_i, is mapped to a interval range. We pass only the label of the data range onto the spatial recognition portion of the recognition system.

Gesture recognition algorithms are generally trained from training data. Most often, the collection of all of the postures, **X**, are used to determine categories of postures, $\mathbf{V} = \{v_1, ..., v_M\}$, that best represent the entire collection of postures [10, 18-21, 27, 33, 42-43, 48-49]. Where M is the total number of posture groups chosen to represent all postures. All of the postures from all of the gestures are used in this step. This is done because there are an immeasurable number of possible postures given slight variations. Each posture group, k_i, is represented by a prototype, $\mathbf{y}_i = \{y_1, y_2, \ldots, y_7)$ of the group and new postures are mapped a the group that they are closest to. The second challenge in gesture recognition involves finding a way to determine the categories of the postures and their associated prototypes. Essentially, it is to find spatial models of the postures.

The third challenge is to find a way to temporally model the gestures as the categories of postures, $\mathbf{V} = \{v_1, ..., v_M\}$, change over the sampled time instances. We now model the gesture as $\mathcal{G}^* = \{v(1), v(2), ..., v(T)\}$. This is generally performed by creating a model of each gesture with respect to how the posture categories change over time.

In the case of this experiment, clustering techniques will be used to model the postures and Hidden Markov Models will be used to represent the postural changes in the gestures.

Figure 14 gives a complete model of the gesture recognition system. It shows the major components and algorithms mentioned above. They are required build the complete gesture recognition system.

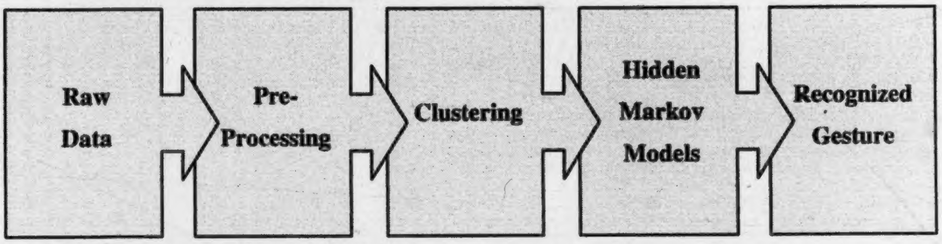


Figure 14 – Major Components of the Gesture Recognitions System

3.3 Examples of Gesture Recognition

There are a number of different applications of gesture recognition. We will now review some of the past experiments. Some types of machine recognised gestures include:

- there have been many attempts to recognize different dialects of sign language [16, 27, 40, 42],
- facial and hand gestures used in video-conferencing [18],
- 3D object manipulation in virtual reality [33],
- written numbers, letters, and shapes in air [48-49],

- integer numbers against a complex background [19],
- head gesture recognition [10],
- gestures related to playing different musical instruments [21],
- and, perhaps the most relevant to this experiment, gestures for the control of mobile robots [20]

There are a number of different algorithms used to recognise gestures. Examples of spatial recognition methods include: various template matching methods [27, 33, 42-43], sub-space models, neural networks [10, 18], clustering [20, 48-49], a multi-principal-distribution-model (PDM) method [19], and an eigen-subspace method [21].

There are also a number of different temporal methods used such as, time-delay RBFNN [17] or other ANN, HMM [10, 19-21, 27, 48-49], and a kinematics based model [28].

There are also a couple of methods that encompass the spatial and temporal modelling into a single algorithm; this includes the use of fuzzy expert systems [16] and multivariate observation sequences for HMM in [35, 40]. [35] also investigates using augmented observation densities in HMM and a condensation algorithm.

3.4 Comparing Gesture Recognition to Speech Recognition

Speech recognition is similar in many ways to gesture recognition. It is the temporal and stochastic nature of speech and gesture signals that make them so closely related. The first stage of speech recognition involves extracting phonemes from the speech. Phonemes are individual sounds that make up words. The second stage involves tracking how the phonemes change over time.

It is interesting to note that just as spoken words are constructed of phonemes, sign language words are constructed of cheremes. Cheremes are different postures made by hands in sign language. In [42], Tamura was able to recognise 20 Japanese sign gestures by matching simple cheremes.

[36] shows, extensively, the successes obtained by applying HMM in speech recognition systems. Given the strong relationship between gesture and speech recognition, it seems natural that HMM should be successful in categorising gestures as well. The primary differences lie in:

- the method of capturing the gestures in a quantifiable way: microphones versus cameras or data gloves and,
- the dimensionality of the problem. Sound varies as amplitude of signal strength and frequency versus time. Gestures are movement in three-dimensional space versus time.

3.5 IR Sensor Based Gesture Recognition

Thus far, gestures have been extracted from space by using one of two methods, visual devices such as cameras or data gloves. In this work, we propose to use 8 infrared sensors to capture the motion of the hands. The sensors are located around the perimeter of the robot and offer a 360° view of hand movement around the robot. See Figure 13 for a bird's eye view of the location of the sensors.

While the sensor said to have a range of 5 cm, the range is limited to about 4 cm from the robot with regards to a human hand. Figure 15 depicts the relationship between the range of an object and the measured value of the IR sensor. Figure 16 depicts the relationship between the palm of a hand and the measured value of the IR sensor. It shows the values of the sensors when a hand is placed pinkie side down perpendicular to the robots central axis at a given distance. In this case the fingers of the hand are slightly closer and the palm of the hand is slightly further away. Note that the measured values are dependent on the current ambient light conditions and that the characteristics of the hand closely resemble those of pink sponge as should be expected since the colour of material closely affects the reflection.

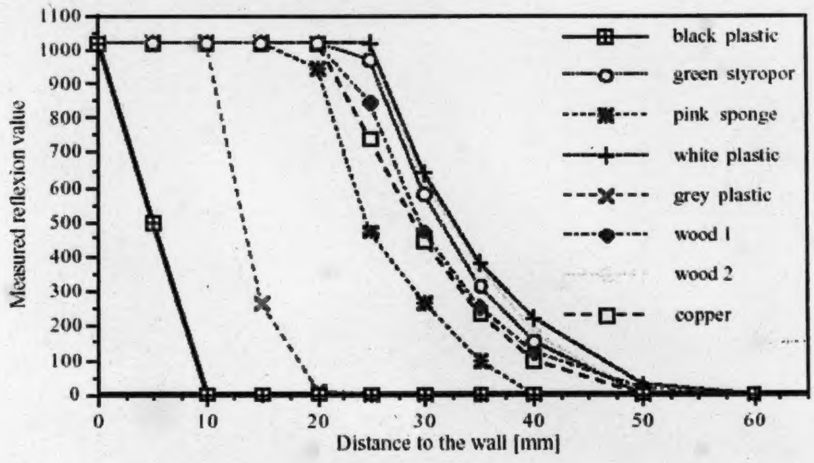


Figure 15 – Measured Reflection Value from IR Sensor vs. Distance to Wall [24]

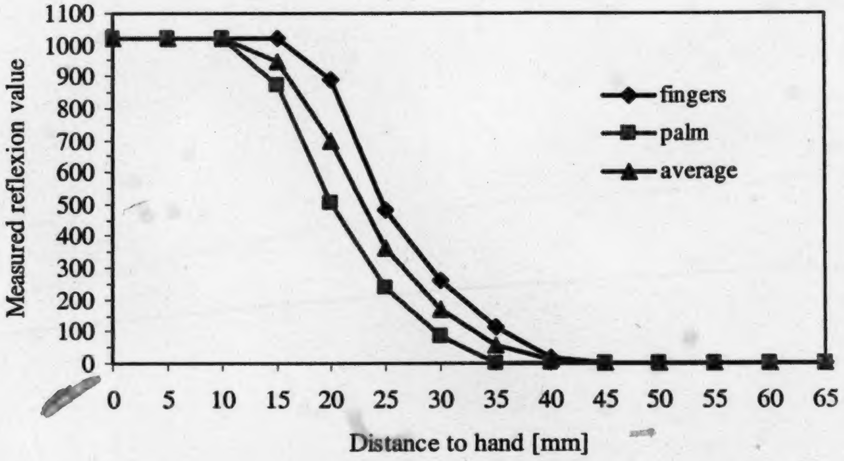


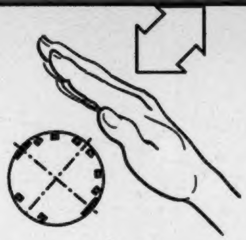
Figure 16 – Measured Reflection Value from IR Sensor vs. Distance to Hand

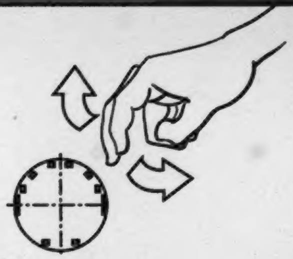
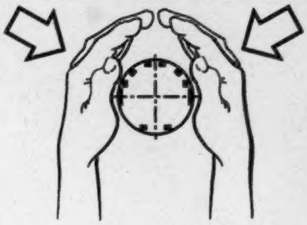
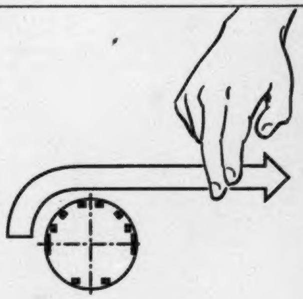
Interestingly, the hand posture is captured mostly by the projection of the hand shape and position onto the robot. This is similar to work in [40] where the silhouettes of hands are captured in images. Only, instead of having a 2 dimensional image, we are limited to an array of 8 sensors with the added information of proximity.

The result is close range detection of gestures and has the following properties. The method offers a very high degree of interaction with the machine and is nearly tactile in nature. By locating sensors on all 360 degrees of the robot, the user is not limited to interactions from one side of the machine. IR sensors also offer the benefit of being very cheap compared to both data gloves and any type of camera input device. They also offer the benefit of flexibility with regards to the number of hands used in a gesture. One or two hands may be used without requiring the addition of another data glove or changes the software to include tracking more than one hand.

3.6 Gestures Recognised

The gestures to be recognised by Hèbert are shown in Table 1 below. Note that all of the gestures are made with the user's hands. Most of the gestures use only one hand or part of it, but some use two.

Subgroup	Diagram	Description and Gestures	#
Hit	 **Figure 17 – Hitting the Robot**	The user moves either hand in the motion of hitting towards the robot. They may hit the front, rear, left or right side of the robot with the palm of the hand facing the robot. Gestures include: *Hit Back*, *Hit Front*, *Hit Left*, and *Hit Right*	4

Subgroup	Diagram	Description and Gestures	#
Tickle	 Figure 18 – Tickling the Robot	The user tickles the robots with one or two fingers. Hèbert is only ticklish in certain places. Front left, front right, and the back. Gestures include: *Tickle Back*, *Tickle Front Left*, and *Tickle Front Right*	3
Capture/ Hug	 Figure 19 – Capturing or Hugging the Robot	The user uses both hands to surround the robot; in essence it does not allow the robot to escape. A capture/hug takes place by moving one hand to the front and one to behind or one hand on to the left and one hand to the right of Hèbert. Gestures include: *Capture From Front and Back* and *Capture From Sides*	2
Direction	 Figure 20 – Directing the Robot to the Right	The user tells the robot to turn 90° and head either to the left or to the right. Either hand begins the gesture on the opposite side that they want the robot to turn and then moves the hand towards the direction of choice. Gestures include: *Go Left* and *Go Right*	2

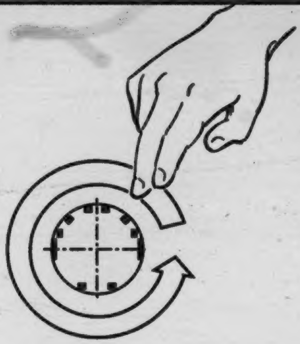
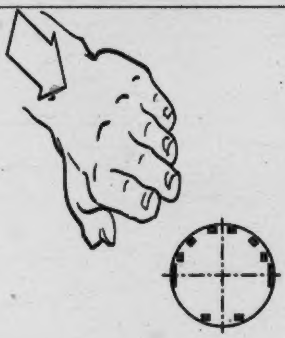
Subgroup	Diagram	Description and Gestures	#
Turn Around	 **Figure 21 – Directing the Robot to Turn Counter Clockwise**	This command requests that the robot turn on its centre. The user uses either hand and circles it around the robot, either clockwise or counter clockwise. Gestures include: *Turn Clockwise* and *Turn Counter-Clockwise*	2
Stay	 **Figure 22 – Directing the Robot to Stay**	This command requests that the robot stop moving. The user slowly moves their hand towards either the front left or front right of the robot. Gestures include: *Stay from Left* and *Stay from Right*	2
Total Number of Gestures			15

Table 1 – Gestures Recognised by Hèbert

The gestures can be grouped into four major categories. The categories represent the type of movement and intention of each gesture. The groups are shown in Figure 23 on the following page.

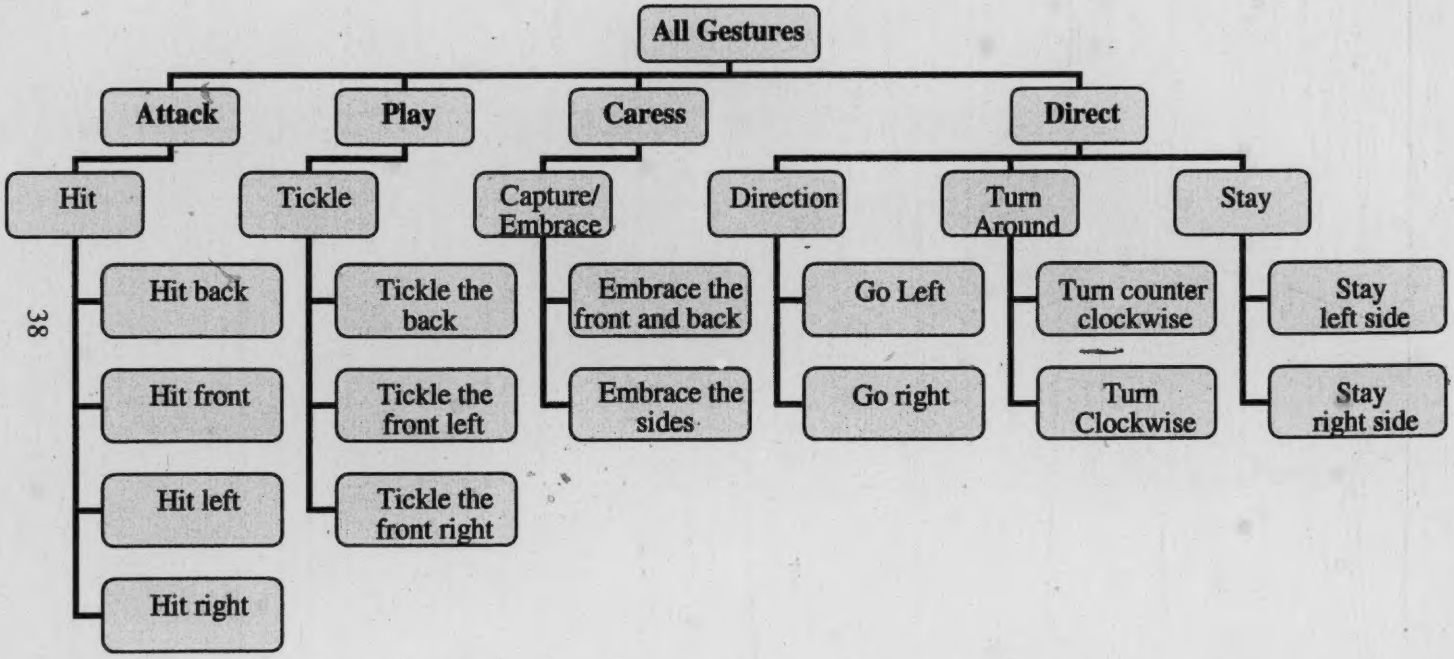

All Gestures
Attack
Play
Caress
Direct
Hit
Tickle
Capture/ Embrace
Direction
Turn Around
Stay
Hit back
Hit front
Hit left
Hit right
Tickle the back
Tickle the front left
Tickle the front right
Embrace the front and back
Embrace the sides
Go Left
Go right
Turn counter clockwise
Turn Clockwise
Stay left side
Stay right side

Figure 23 - Categories of Gestures

3.7 Variations in Gesticulation

It is interesting to note how Figure 12 clearly shows three different ways people gesticulate beckoning. Of course, there are countless other ways. One of the challenges in creating a gesture recognition system is to find a balance between training users and allowing flexibility in the models. Too much human training removes some of the naturalness of gestures, but there is a limit to how much flexibility can be built into a model without retraining it. Even people need to learn a new set of gestures when they visit different cultures.

As such, the volunteers were given a brief demonstration of how to perform each gesture, but they were given freedom to perform them as they felt most comfortable and natural. In other words, each participant held their hands in slightly different positions and gesticulated at different speeds.

3.8 Data Collection of Human Gestures

We need to determine a way to collect the gestures. Factors to consider include: where to perform the gestures, when to start tracking gestures, how long to continue tracking gestures, how often to sample the sensors (sampling rate), who should perform the training and testing gestures, and how to divide the data samples.

3.8.1 Where to Perform the Gestures

Let us begin by considering where to perform the gestures. Recall that the sensors have at most a 4 cm range with regards to a human hand. This means that the gestures must be performed within at most 4 cm of the robot. The robot has 6 sensors along the front and two at the rear as shown in Figure 13; they are reasonably well distributed around its perimeter. There are no sensors on either the top or the bottom of the robot. As such, the user must gesticulate around the perimeter of the robot.

3.8.2 When to Start Tracking Gestures

There is also the challenge of trying to figure out when to begin sampling the gestures. This could be done continually, but, if it is then, we need to develop some way to extract the gestures from the continuous stream of data. Given that the gestures are performed so close to the robot and that hands are not normally placed within the range, it is easiest to begin tracking the gestures only when something is detected within the range of the sensors. In other words, when an object is detected within the range of the sensors, it triggers the robot to begin sampling. The user can delimit an interaction session by capturing the robot; this ensures that the robot does not confuse encountering a wall with a human gesture.

The 10-bit sampling of the sensors produces values of 0-1023. 1023 represents an object nearly touching or touching the sensors. Because of slight fluctuations due to noise, both 0 and 1 are the readings when nothing is detected within the sensor range. As such, the sum value of all of the sensor reading must be greater than 8 before the tracking (recording) of the gestures begins.

3.8.3 Duration of Sampling

Since we are not sampling continuously we need to determine how long to keep recording the sensor values once we have started. The duration of sampling relates directly with the length of time it takes to perform the gestures. The longest gesture lasts for about one second. All gestures will be sample for a second. Shorter gestures will simply be padded with the low readings of the sensors.

3.8.4 Sampling Rate

The next question to consider is, how often to sample the data? The key here is to sample the data at a rate that is just capable of capturing sufficient information. Because we want to recognize the gestures in real time, we want to minimise the amount of data to process. This can be done by capturing as little data as possible. A number of samples were taken with various sampling rates. The sampling rate of 13 samples per second appeared to produce data with sufficient information. It is interesting to note that this value is also very close to the minimum frame rate used in animation to sustain the human perception of motion.

3.8.5 Collecting Training and Testing Data

A number of samples of gestures must be collected in order to model the gestures for the recognition system. The goal is also to have the models to be as general as possible. In other words that a gesture should be recognizable regardless of who has performed it.

Four different people performed each gestures 40 times. As described above, the gestures were captured as soon as the user moved their hand within range of the sensors and each one was captured for one second.

The resulting data set contained 4 times 40 copies of each of 15 gestures for a total of 2400 gestures. The number of each of the gestures to record is based on previous experiments [48-49].

3.9 Chapter Summary

Gestures are a natural method of communication; they are movements of the body that convey information or desires and they can be intentional or non-intentional. They are often represented as postures that change with time.

There have been a number of experiments relating to gesture recognition, including interaction with robots. There are a number of different approaches to gesture recognition. Some of them extract postures first and then develop a temporal model while others use methods that model the spatial-temporal elements together. Many examples have shown that gesture recognition systems using HMM have been successful. Gesture recognition also has some similarities to speech recognition where HMM have been used quite successfully to model spoken language.

15 different gestures are recognized in this experiment. These gestures can be performed by either hand or both hands at once. They can be broken down into 4 categories: "Attack", "Play", "Caress", and "Direct". There are also subcategories within each of the categories.

Careful consideration was given to how to collect gesture data and represent it in a numerical form that can be used to both model and classify the gestures. The result is that the gestures are made about the perimeter of the robot and are sampled 13 times per second. Each gesture is recorded for 1 second and recording begins as soon as an object is detected within any of the sensors range. A user delimits their interaction session by greeting the robot with a hug.

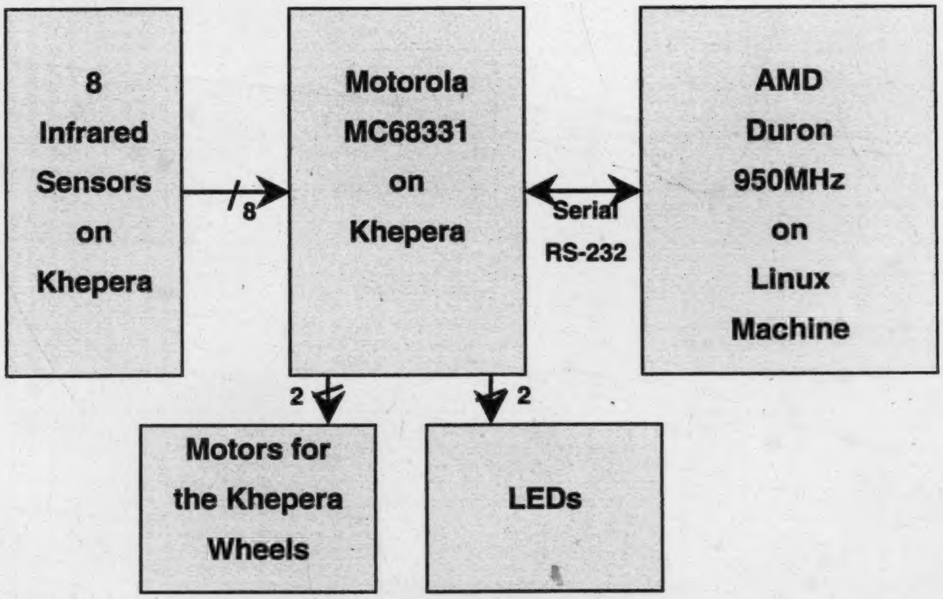
Four people performed each of the gestures 40 times. This produced a data set of 2400 gestures. This data set was then parsed into two groups, one for testing and one for training.

Chapter 4: Implementation of a Human Robot Interface

4.1 Overview

The goal of this chapter is to present the hardware architecture and the data path for the implementation of the interactive robotic system. We begin by presenting the hardware architecture and then describe the data flow for the three different phases of the system.

4.2 Hardware Architecture



8 Infrared Sensors on Khepera
8
Motorola MC68331 on Khepera
Serial RS-232
AMD Duron 950MHz on Linux Machine
2
2
Motors for the Khepera Wheels
LEDs

Figure 24 – Hardware Architecture

The hardware of the recognition system has two major components: the Khepera robot and an AMD Linux machine. The Khepera robot has 8 infrared sensors, two motor controlled wheels, two LEDs, and a Motorola 68HC331 onboard. This robot connects to the AMD 950MHz based Linux machine via an RS232 serial line. Figure 24 represents the hardware architecture and data flow between the relevant components.

The roles of each component follow:

- The **8 Infrared Sensors** on the Khepera robot capture hand posture and position.
- The **Motors** for the Khepera drive wheels according to the behaviour model.
- The **LEDs** on the Khepera represent the emotional state of the robot.
- The **Motorola 68HC331 on the Khepera** samples the information. The microprocessor samples the data approximately 13 times a second. It formats the data and sends it over an RS-232 serial line to the AMD Duron on the Linux Machine.
- The **AMD Duron** on the Linux machine performs the remaining processing of the data.
    - In the capturing stage, it records the data.
    - During the training and testing stages, it processes the data and forms and verifies the models.
    - During the recognition stage, it processes the data, then identifies the gesture, determines the robot behaviour, and then passes desired behaviour information to the robot.

4.3 Data Processing Flow

The data flow through the recognition system is broken down into three different phases. The first phase is the training stage, followed by the testing phase, and later the interaction phase.

It is necessary to divide the sample data set into two separate groups. One group will be used for training and the other will be used for testing. Every second gesture was placed into either the training or the testing group. This was done to ensure that an equal number of gestures from each person were placed into both groups. The same method is used in [48-49].

4.3.1 Training Phase

The recognition system must undergo training before it can identify the gestures. Figure 25 depicts the flow of the data through the steps required for the training process.

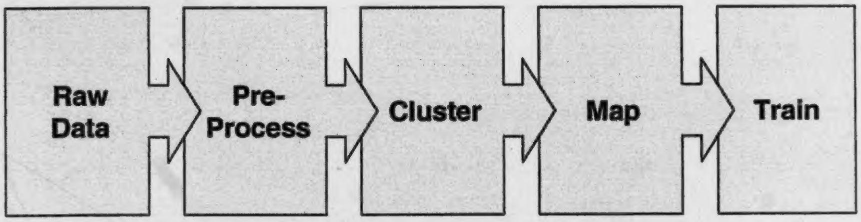


Figure 25 – Data Processing Through Training Steps

The steps in Figure 25 are explained in more detail below:

- **Raw sensor values** range from 0-1023. The microprocessor on board the robot samples the sensors 13 times per second.
- **Pre-processing** involves quantisation of the individual sensor values to one of n intervals
- **Clustering** involves finding posture group prototypes from the 8-dimensional input vectors in the training data. We then construct the codebook of the HMMs from these posture group prototypes. The codebook is the alphabet of observations presented to the models.
- **Mapping** involves assigning each instance (set of 8 sensor values) of the data set to one of the prototypes. This stage produces usable observation sequences for use in training the HMMs.
- **Training** involves generating the HMMs. We train the models using the mapped data.

4.3.2 Testing Phase

After the training stage is complete, testing is required to ensure that the system can correctly identify the gestures. Figure 26 outlines the data flow through the basic steps of the testing stage. Note that the first steps of the testing stage are the same as those of the training phase.

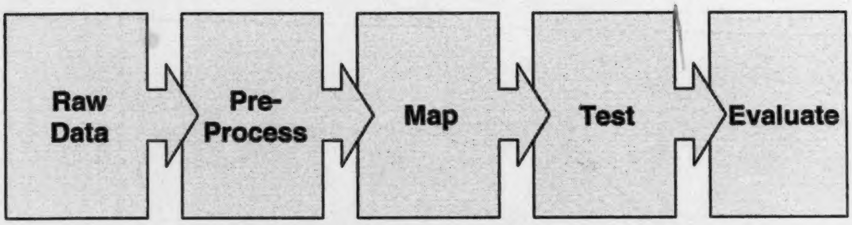


Figure 26 – Data Processing Through Testing Steps

The details below further explain the steps in Figure 26 that were not described previously:

- **Mapping** involves assigning each instance (set of 8 sensor values) of the data set to one of the prototypes found in the training stage. This stage produces observation sequences used for testing the existing HMMs.
- **Testing** involves taking the observation sequences and comparing them to each of the models. We find the probability that the observation sequence belongs to a given HMM. The HMM that generates the highest probability of the observation sequence given the model is chosen as the probable gesture.
- **Evaluation** involves taking the data that results from the testing stage and investigating the accuracy of the models. The data evaluated includes misclassifications of gestures and frequency of

mapping to a prototypes from the clustering step. It involves identifying problem areas and determining potential improvements.

4.3.3 Real Time Interaction Phase

Once we validate the posture group prototypes and the HMMs in the testing phase, the system is ready to perform real-time recognition and interaction. The first steps of the data flow through this stage are the same as in the testing stage. Figure 27 outlines the steps.

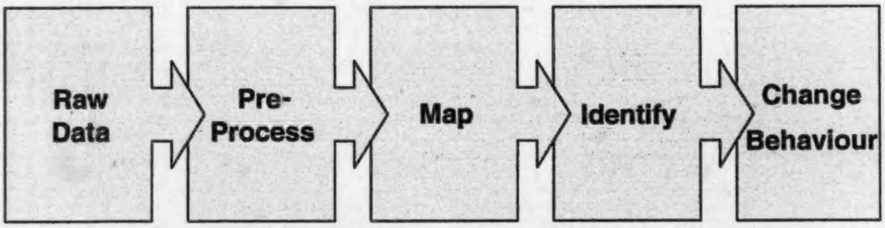


Figure 27 – Data Processing Real-Time Recognition Steps

The details below further explain the steps in Figure 27 that are not explained above:

- **Identification** is the same as the testing step in the testing stage. It involves taking the observation sequences and comparing them to each of the models. We find the probability that the observation sequence belongs to a given HMM. The HMM that generates the highest probability of the observation sequence given the model is chosen as the probable gesture.
- **Changing Behaviour** involves determining the current emotional state and behaviour of the robot based on the history of identified gestures. The desired actions of the robot are passed from the AMD machine to the robot; it modifies the action motor, and LED drives accordingly.

4.4 Chapter Summary

In this chapter, we presented the hardware architecture and the data paths of the three different phases of the system. We saw that the key components of the system are the robot sensors and actuators, the microprocessor on board the robot, and the AMD Duron Linux based computer. We then presented the data processing flow of the three different phases of the system: training, testing, and real time interaction.

Chapter 5: Pre-Processing

We will now consider the role of pre-processing in a recognition system. We will begin by presenting some background information about pre-processing. We will then present how pre-processing was actually implemented in this experiment.

5.1 Background

The current goal of pre-processing is two-fold. First, we want to extract the most relevant information from the data and second we want to reduce the level of complexity of the data. In many cases of gesture recognition pre-processing involves tracking hands, extracting angle and velocity information from images, or sensor glove data.

Due to the simple nature of the IR sensors, the data here is already reasonably simple. We want to be able to recognize gestures from one or two hands. If we considered a hand to project onto the IR sensors, we could extract information such as the center of the data projection spread and the width of the projection, or the closest point of the spread. The problem is that since we want to model two hands instead of just one, we would need this information twice. The amount of data that it would take to represent this information is hardly less than if we simply left the sensor information as it was. This being said there is still some pre-processing that would be beneficial.

5.2 Actual Pre-Processing

The range of the sensors is limited but the resolution is high. The 10 bits of resolution is higher than the precision of hand positions and movement within the sensor range. Therefore, it would be more useful to break the information provided by the sensors into fewer categories such as: absent, near, very near, and extremely near or 1, 2, 3, and 4.

There are two main benefits of performing this type of pre-processing. The first is that it allows more flexibility in hand position and the second is that it reduces the complexity of the data presented to the clustering algorithm.

The challenge is to select appropriate boundaries to the category intervals and to determine if the intervals should be determined for each sensor separately or it they can share the same boundaries. Considering that the same gestures are performed at different locations around the robot, it is assumed that the sensors all receive very similar input. Hence, the boundaries will be common to all sensors.

In order to determine where to draw the boundaries of these categories the data set was examined. The occurrence of every value from 0-1023 was counted and histograms were created from the entire data set.

The first histogram, Figure 28, shows all of the data, where each bar of the histogram represents a group of sensor values. Generally, each bar of the histogram represents an interval of 25 sensor values with the exception of the first two and the last two groups. The actual groups follow:

0-1, 2-25, 26-50, 51-75, ..., 976-1000, 1001-1022, 1023

This division was used because of the high number of occurrences of the numbers 0 and 1 and 1023. 0 and 1 represent the case where the users hand is not detected by the sensor and 1023 is the maximum value and represents any case where the hand is close enough to the sensor to produce this value. For additional clarity a histogram is shown in Figure 29 that excludes the extremity groups, 0-1 and 1023.

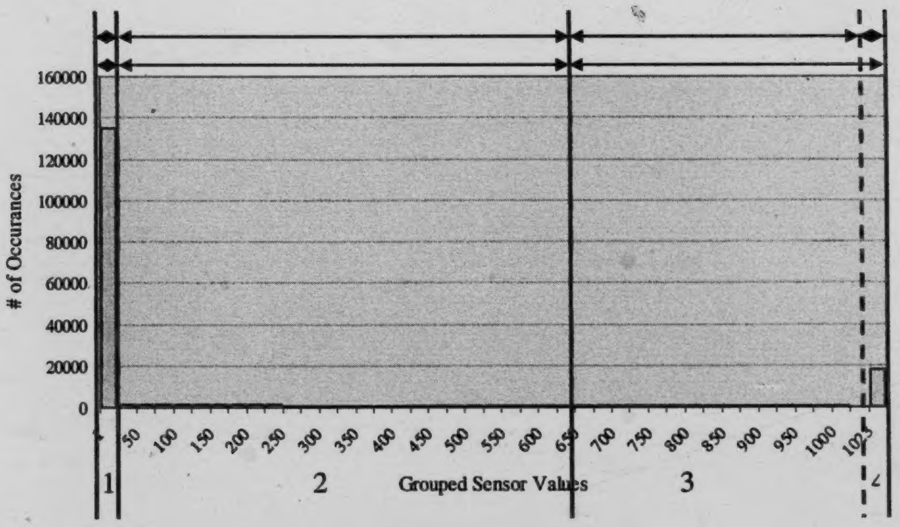


Figure 28 – Distribution of the Occurrence of Grouped Sensor Values Including Extremities

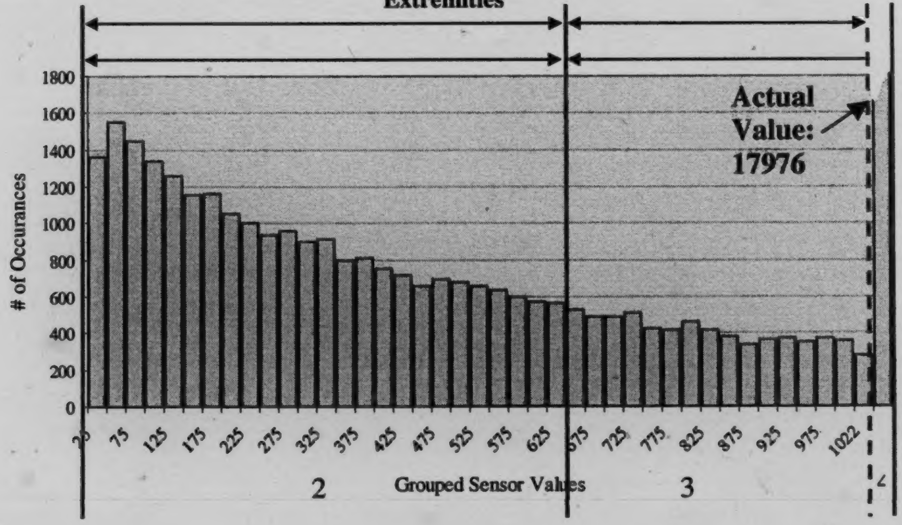


Figure 29 – Distribution of the Occurrence of Grouped Sensor Values Excluding Extremities

5.2.1 Two Approaches to Assigning Intervals

5.2.1.1 Three Intervals

From the distribution, it is obvious that the group of 0-1 should be assigned an interval of its own. These are the values that are produced when the sensors detect nothing; we will interval 1 "**absent**". The remaining values could be broken down into two more intervals. The two remaining groups were assigned as follows: interval 2, "**near**", contains all sensor values from 2 up to 649 and interval 3, "**very near**", contains all sensor values 650 and above. There are two major reasons for this split; the first is the distribution of the data and the second is the relationship between hand distance and sensor values. According to the cumulative distribution, when we exclude the first interval, approximately half of the data fits into interval 2 and the other half into interval 3. When you consider the relationship between the proximity of the hand and the value of the sensor, the value of 650 is represents a hand that is approximately 20mm from the sensor, this is the midpoint of the range. Figure 28 shows the division of the intervals. Note that the solid lines denote the divisions for the 3 intervals, the dashed line is used when dividing the data into 4 intervals.

5.2.1.2 Four Intervals

Another option was examined with the case of assigning sensor values to 4 intervals. The main difference here is that interval 4 is labelled "**extremely near**"; it represents the value of 1023. Interval 3, labelled "**very near**", contains sensor values from 650 to 1022. Note the divisions marked on Figure 28, the dashed line represents the split of interval 3 into the two new intervals.

5.3 Chapter Summary

Pre-processing is used to extract relevant information from data thereby reducing the amount of processing in subsequent steps. In the case of this experiment, raw sensor measurements are assigned to one of three or four intervals. The intervals represented the states of the hand being: absent, near, very near, or extremely near. The limits of the intervals were determined by analysing the entire collection of raw sensor measurements and plotting histograms. The intervals limits were then assigned according to the distribution of the data.

Chapter 6: Clustering Techniques

In this chapter, we present clustering techniques. We begin by explaining what clustering is and what it attempts to do. We then examine the basic categories of clustering algorithms. Following that, we look at how to select a clustering method from the basic categories. Specific algorithms of objective functional clustering and their associated parameters are presented next, namely K-Means and Fuzzy C-Means. We also compare them against each other. Finally, we present how the selected clustering algorithm is actually implemented in this experiment.

6.1 Background

Our current model of gesture recognition categorises spatial information first by identifying posture groups and then categorises temporal changes in the posture groups. Clustering is used to generate a discrete number of posture groups that we can map inputs to before the HMMs are trained and later used for tracking the temporal changes in the groups. We use only the training portion of the data set when attempting to find the clusters. Recall that it is necessary to divide the sample data set into testing and training groups.

Clustering is a process that builds a structure from data and the structure can then be used to assign data points to a particular category according to the features chosen. It is useful in this application because clusters can be formed from the training data set, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. Where each $\mathbf{x}_i$ is a vector of all the sensors of the robot. The entire data set is comprised of all of the input vectors from all of the gestures concatenated together. The prototypes, $\mathbf{y}_j$, of these clusters can then be used to classify new input vectors, $\mathbf{x}_i$, from the entire data space. In other words, the prototypes are used to map incoming data to an appropriate label that represents the hand posture from the robot sensors.

These labels are then passed on as input to the temporal classification stage, the HMM.

So what exactly are clustering algorithms? They are a subset of pattern recognition algorithms and belong to the category of unsupervised learning algorithms. Unsupervised learning algorithms are useful for finding something of value in unlabeled samples and clustering methods seek similarities between subsets of a sample. Duda states, "*clustering procedures yield a data description in terms of clusters of groups of data points that possess strong internal similarities* [13]."

6.2 Categories of Clustering Algorithms

There are a number of different clustering procedures, and not one of them is universally applicable. Depending on the nature of the data and the desired application of the clusters, there are methods that are more suited to the problem. There are three main categories of clustering algorithms: hierarchical methods, graph theoretic methods, and objective function methods.

6.2.1 Hierarchical Methods

Hierarchical methods are agglomerative (merging) or divisive (splitting) in nature. The application of the algorithms yields a hierarchy of nested clusters. The methods are suitable for data structures that are dendritic in nature and are frequently used in taxonomical studies. An advantage of these methods is that they are both computationally and conceptually simple.

6.2.1.1 Agglomerative Methods

In the agglomerative methods, we begin with each data point forming its own cluster. We then search for similarities, based on a selected criterion, between one clusters and the other clusters and then merge similar clusters together. This process can continue until all of the clusters are agglomerated into one large group containing all data.

Take, for example, a garden. We would label each type of plant and put them into their own cluster. Then we could take all plants with similar characteristics and merge their clusters into an encompassing cluster. New clusters that contain the first set of clusters could be plants that produced root vegetables, plants that climb up trellises, short flowering plants bearing no fruit, and short flowering plants that produce fruit. From this point, a new set of clusters could be ornamental plants and food-producing plants. Finally a cluster that represents plants could be formed. See the following diagram for details.

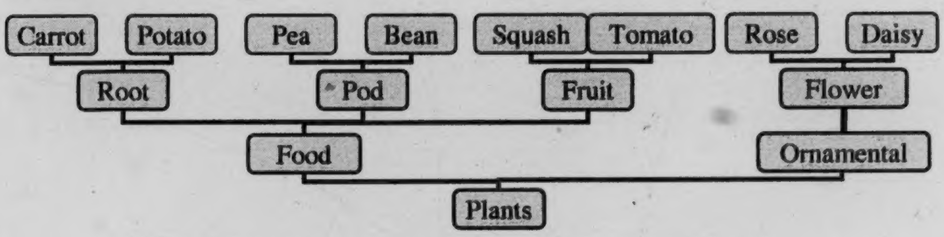


Figure 30 – Example of Agglomerative Clustering Methods

6.2.1.2 Divisive Method

The divisive method of hierarchical clustering approaches the problem the opposite manner than agglomerative clustering. We begin with one large cluster representing all of our data. We then begin to split the data, according to a selected criterion, into two or more clusters at each step. We can continue up until we only have one data point in each cluster.

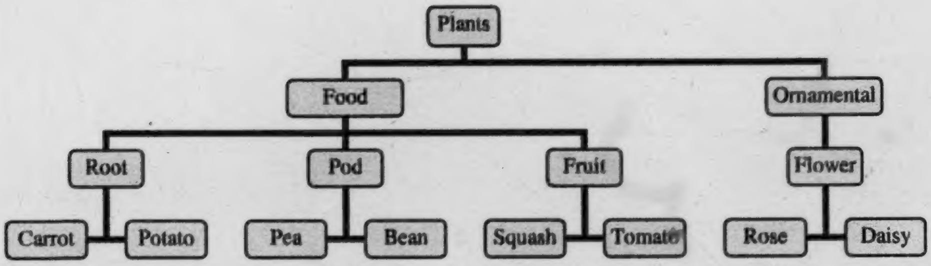


Figure 31 – Example of Divisive Clustering Method

Let us revisit the garden from above. Using these methods, we would begin with the category, plants, to represent all living, non-animal, items in the garden. Next we could make a split according to whether or not the plants were ornamental or food-producing. From within each of these clusters we could now split the data into height clusters. From within each of those clusters we could split them into flower or non-flower producing plants. This process could continue until all plants were represented by their own cluster. See the following diagram for details.

6.2.2 Graph Theoretic Methods

Clusters can also be found in data using graph theoretic methods. The data points in **X** are represented as separate nodes. Similarity measures, edge weights, are taken between pairs of nodes. When the similarity measure is determined to be too small, the link between the two nodes is broken. In other words, links are broken when the link between two nodes is longer than a threshold amount. The groups of nodes with unbroken links form clusters, or sub-graphs.

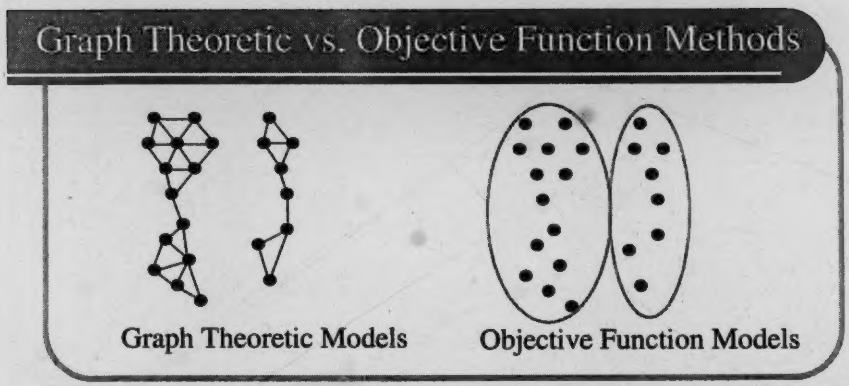

Graph Theoretic vs. Objective Function Methods
Graph Theoretic Models
Objective Function Models

Figure 32 – Comparing Clusters from Graph Theoretic and Objective Function Methods [3]

These methods are well suited to data with a chain or pseudo-linear structure but not to hyper-elliptical clusters, or data with noise or bridges. They are also unsuitable in applications that require a prototype classifier because there are no paradigmatic representatives generated for each subclass. Figure 32 shows an example of a data structure well suited to graph theoretic methods.

6.2.3 Objective Function Methods

This brings us to our third category, objective function methods. Objective function methods use a performance index, *J*, to evaluate the "fitness" of the clusters. A basic example of a performance index is given in equation 5.1.

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| \mathbf{x}_i - \mathbf{y}_j \right\|^2 \qquad \textbf{(6.1)}$$

Where

$$\mathbf{x}_i, 1 \le i \le N; \qquad \textbf{(6.2)}$$

is a input vector and

$$\mathbf{y}_j, 1 \le j \le k; \qquad (6.3)$$

represents a cluster prototype or center. N is the total number of data vectors and k is the total number of clusters.

$$\left\| \mathbf{x}_i - \mathbf{y}_j \right\| \qquad (6.4)$$

is any distance function.

These algorithms attempt to find an extreme of the performance function in order to find the best fit for the clusters. The function generally contains some type of similarity or dissimilarity measurement. These measures are most often a type of distance measurement.

While an exhaustive search of the solution space is possible, generally, the algorithms iteratively update clusters based on an update formula until a stopping criterion is reached. This is because exhaustive enumeration is computationally prohibitive. The unfortunate result of iterative optimization is that the solution is locally rather than globally optimised. Objective function methods are said to be the most precise method of clustering, although high "precision" does not necessarily result in the best clusters.

Objective function methods are useful in applications where the data is hyper-elliptical, or some other hyper-shape, in nature. For best results, the data should be fairly evenly distributed amongst the generated clusters and no outliers should exist in the sample data. These methods are also applicable to classifier systems because the solution yields prototypes for each cluster.

Details concerning the implementation of an objective function method of clustering will follow in section 4.3.

6.3 Selecting a Clustering Method

After careful consideration of the various clustering methods, it seems that an objective function technique would be best suited to the problem presented in this thesis. The main characteristics of the three clustering methods are summarised in Table 2 The main reason for choosing this method is based on the nature of the data. The data is neither dendritic, nor is it chain-like or pseudo-linear. It will be assumed that the data in this study has a hyper-elliptical structure and that it contains noise. A further reason for selecting an objective functional method is that a desired outcome of the clustering procedure is a classifier system. The generation of cluster prototypes is essential to the implementation of this gesture recognition problem.

Feature	Hierarchical/ Graph Theoretic	Objective Functional
Data Structure	Dendritic, Easily linked by chains	Hyper-shaped clusters
Cluster Prototypes	Hard to find	Easily extractable
Theory	Simple	Simple

Table 2 – Comparing Clustering Methods

6.4 Specific Algorithms of Objective Functional Clustering

There are countless implementations and variations of objective functional clustering. Therefore, while it is useful to have narrowed down the selection of methods, there are still decisions left to make concerning the specific algorithm. We will review two of the methods: k-means clustering and fuzzy-c means, FCM, clustering.

6.4.1 K-Means Clustering

In order to gain a better understanding of the how objective functional clustering works, we will first review its simplest implementation: K-means clustering. Different versions of k-means clustering were developed in many nations in the 1960's. It was first partly published in 1967 as (sequential) k-Means by MacQueen [13].

6.4.1.1 The K-Means Algorithm

The method works by first selecting the desired number centres or clusters, k, where $k<N$. The next step is to initialise the centres, $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$. Following that, each input vector, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ is assigned to a centre, $\mathbf{y}_j$, by evaluating a selected distance function. The data points are assigned to the cluster that they are closest to. The next step is to update the cluster centres by calculating the new means. The objective function, J, is then calculated and, most often, compared to the previous iterations value. Two main stopping criteria are possible:

1. stopping after a set number of iterations are completed, or
2. more commonly the procedure terminates once the change in J is smaller than a previously selected threshold amount, ε.

A typical implementation of the k-means algorithm is shown in Table 3 [15].

K-Means Algorithm	
1) Initialisation	Choose a set of cluster centres (prototypes) arbitrarily: $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k\}$
2) Similarity Matching	Assign samples, $\mathbf{x}_i$, to cluster, $\mathbf{y}_j$, using Euclidean distance: $\mathbf{x}_i$ belongs to cluster $\mathbf{y}_j$ if $\|\mathbf{x}_i - \mathbf{y}_j\| < \|\mathbf{x}_i - \mathbf{y}_l\|, \quad j \neq l \quad \forall j$ **(6.5)**
3) Updating	Compute the mean value of all data points in the cluster: $\mathbf{y}_j = \sum_{i \in S} \frac{\mathbf{x}_i}{\|S\|} \quad \forall j$ **(6.6)**
4) Continuation	Repeat steps 2 and 3 until there is a small change in the cost function: $J = \sum_{j=1}^{k} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{y}_j\|^2 < \varepsilon$ **(6.7)**

Table 3 – K-Means Algorithm

6.4.1.2 Selecting parameters

We need to determine a number of parameters, in order to successfully implement the k-means algorithm. They include:

- selecting the number of centres, k,
- selecting a method to compute the centroids or means of the clusters,
- initialization of centres, $\mathbf{y}_j$,
- selecting a distance function,
- determining the stopping criterion and/or threshold, ε.

Number of Clusters

Given any data set, desired application, and clustering method there exist an optimal number of clusters. Nevertheless, the question remains; how do we determine this number? To date, there appears to be no theoretical method by which this number can be calculated. The individual must consider the data and application and then attempt to select an appropriate number of clusters. The literature suggests empirically determining this number. That is to say, one should repeat the clustering procedure for different number of clusters, and then select the appropriate number of clusters according to the results obtained.

Computing the Centroids

There are three categories of representation of data to consider when determining which method to use to calculate the centroids of the cluster centres. The first is when the data, X, is represented quantitatively according to the entity to variable relationship. In this case, a centre of gravity method can be used:

$$\mathbf{y}_j = \sum_{i \in S} \frac{\mathbf{x}_i}{|S|} \qquad \textbf{(6.8)}$$

The second is when the data is represented nominally. In this case, a vector of the modal categories of the variables in X can be used. The final case is when the data is represented in terms of a dissimilarity matrix. In this case, centroids are computed as those that minimises the total dissimilarity within the clusters. Only the details relating to the centre of gravity method are explained. For more information on other data representations, please refer to [13].

Initialisation of the Centres

The next question to consider is, how to initialise the centroids of the clusters? The simplest method is to assign the values randomly. Other alternatives include: randomly selecting the centres from the data set, using a threshold method, or expert selection [13]. The random method is self-explanatory. The threshold method involves selecting the centres from the data set. A new centre is assigned once a data point is further from all the existing centres than a predetermined threshold amount. Expert selection involves employing the users understanding or knowledge of the problem to determine initial values for the centres.

Distance Measure

One very important aspect of an objective function method is the measure used to determine the similarity between input vectors, **x**, and the cluster centres, **y**. The most direct and commonly used similarity, or dissimilarity, measure employed is a distance measurement [11]. There are a number of different distance measurements from which to choose. Essentially, each distance measurement takes on a particular shape in n-dimensional data space. Take for example the Minkowski distances. The Minkowski distances are defined by:

$$\|\mathbf{x}-\mathbf{y}\| = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|} \qquad (6.9)$$

Both **x**, **y** $\in \Re^n$. This equation represents an infinite family of distance measures that vary with the definition of the variable $p \geq 1$.

Some of the most common distance functions include:

- the Hamming distance, where $p = 1$;

$$\|\mathbf{x}-\mathbf{y}\| = \sum_{i=1}^{n} |x_i - y_i| \qquad (6.10)$$

- the Euclidean distance, where p = 2;

$$\|\mathbf{x}-\mathbf{y}\| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|} \qquad (6.11)$$

- and the Tschebyshev distance where p = ∞;

$$\|\mathbf{x}-\mathbf{y}\| = \max_{i=1,2,\ldots,n} |x_i - y_i| \qquad (6.12)$$

To gain a better understanding of the selection of a distance function, let us examine the case where the dimensionality of the data is limited to 2. Figure 33 shows the space that the distance functions interpret. The Hamming distance produces the diamond shape, the Euclidean the circle, and the Tschebyshev yields the square [11]

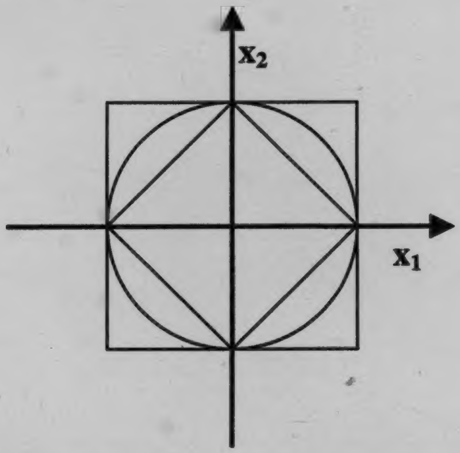


Figure 33 – Examples of Distance Functions from Minkowski Distance Family [11]

Further to the infinite class of Minkowski distance there is also the family of Mahalanobis distances. It is defined by:

$$\|\mathbf{x}-\mathbf{y}\| = (\mathbf{x}-\mathbf{y})^T \mathbf{M}^{-1} (\mathbf{x}-\mathbf{y}) \qquad (6.13)$$

where M is a positive definite matrix. The advantage of this distance measurement is that it represents the distance between individual features very well. When M is a diagonal matrix with its elements equal to σ_i^2,

$$\mathbf{M}^{-1} = \begin{bmatrix} \sigma_1^{-2} & 0\ldots & 0 \\ 0 & \sigma_2^{-2}\ldots & 0 \\ 0 & 0\ldots & \sigma_n^{-2} \end{bmatrix} \qquad (6.14)$$

we get the Bhattaharaya distance:

$$\|\mathbf{x}-\mathbf{y}\| = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\sigma_i^2} \qquad (6.15)$$

A special case of the Bhattaharaya distance occurs when M is the identity matrix. In this case the distance function reduces to a regular Euclidean distance:

$$\|\mathbf{x}-\mathbf{y}\| = (\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y}) \qquad (6.16)$$

The Euclidean distance is chosen as our distance function because it is a member of both families of distances, as such; it shares the advantages of each family. It also represents a hyper-elliptical shape, which makes it invariant to rotations and translations. Note that we are assuming that our clusters are of that shape.

Stopping Criterion

Finally, there is the stopping criterion to consider. There are two main ways to stop the algorithm: the first one is by stopping when the change in the objective function is smaller than a certain threshold or secondly by setting the maximum number of iterations. These criteria can be used alone or together. For instance, one could set the threshold of change to a certain level and then as backup set the maximum number of iterations. This is useful in case the objective function begins to oscillate and does not meet the first criterion. The selection of these values is dependent on the nature of the problem and the data. If the value of the objective function is large, the threshold value should be larger than if the value of the objective function is small.

6.4.2 Fuzzy C-Means Clustering

Fuzzy C-Means, FCM, is an extension of the basic k-means algorithm. The major difference is that, instead of a data point, $\mathbf{x}_i$, being assigned completely to one cluster, $\mathbf{y}_j$, it is assigned a membership value, u_{ij}, to the cluster. Where $u_{ij} \in [0,1]$.The goal is to minimise the objective function [3, 34], defined as:

$$J_m(\mathbf{U},\mathbf{y}) = \sum_{j=1}^{k}\sum_{i=1}^{n}(u_{ij})^m \left\|\mathbf{x}_i - \mathbf{y}_j\right\|^2 \qquad (6.17)$$

Where $\mathbf{U} \in \mathbf{M}_{fc}$ is the fuzzy partition matrix and $m \in [1,\infty)$ is the weighting exponent of the partition matrix.

In order to define the rules for the membership values, let us begin by defining, ∀i, the sets:

$$\mathbf{L}_i = \{ j \mid 1 \le j \le k; \left\|\mathbf{x}_i - \mathbf{y}_j\right\| = 0 \} \qquad (6.18)$$

$$\widetilde{\mathbf{L}}_i = \{1, 2, \dots, c\} - \mathbf{L}_i \qquad (6.19)$$

We can then calculate the values of u_{ij} as follows:

$$\mathbf{L}_i = \emptyset \Rightarrow u_{ij} = \frac{1}{\left[\sum_{l=1}^{k}\left(\frac{\left\|\mathbf{x}_i - \mathbf{y}_j\right\|}{\left\|\mathbf{x}_i - \mathbf{y}_l\right\|}\right)^{2/(m-1)}\right]} \qquad (6.20)$$

or

$$\mathbf{L}_i \neq \emptyset \Rightarrow u_{ij} = 0 \ \forall j \in \tilde{\mathbf{L}}_i \quad \text{and} \quad \sum_{i \in \mathbf{L}_i} u_{ij} = 1 \qquad (6.21)$$

The means of the cluster prototypes are calculated by:

$$\mathbf{y}_j = \frac{\sum_{i=1}^{N} (u_{ij})^m x_i}{\sum_{i=1}^{N} (u_{ij})^m} \quad \forall j \qquad (6.22)$$

Note the similarity the calculation to the center of gravity method above.

6.4.2.1 The FCM Algorithm

The FCM algorithm begins by selecting a value for *m* and initialising the fuzzy c-partition membership matrix, $\mathbf{U}_0 \in \mathbf{M}_{fc}$. This is typically done by randomly assigning values of the matrix. The next step is to compute the means of each cluster. This is computed according to equation 5.22. Following that, the partition matrix is updated according to equation 5.20 or 5.21. The procedure continues until the stopping criterion is met.

FCM Algorithm	
1) Initialisation	Randomly initialise the fuzzy partition matrix $\mathbf{U}_0$ and fix *m*.
2) Compute Means	Compute the means of the cluster prototypes according to: $\mathbf{y}_j = \frac{\sum_{i=1}^{N} (u_{ij})^m x_i}{\sum_{i=1}^{N} (u_{ij})^m} \quad \forall j$ **(6.23)**
3) Updating	Update the fuzzy partition matrix, **U**, according to: $\mathbf{L}_i = \emptyset \Rightarrow u_{ij} = \frac{1}{\left[\sum_{l=1}^{k} \left(\frac{\lVert \mathbf{x}_i - \mathbf{y}_j \rVert}{\lVert \mathbf{x}_l - \mathbf{y}_j \rVert} \right)^{2/(m-1)} \right]}$ **(6.24)** or $\mathbf{L}_i \neq \emptyset \Rightarrow u_{ij} = 0 \ \forall j \in \tilde{\mathbf{L}}_i \quad \text{and} \quad \sum_{i \in \mathbf{L}_i} u_{ij} = 1$ **(6.25)**
4) Continuation	Repeat steps 2 and 3 until there is a small change in the cost function: $J_m(\mathbf{U}, \mathbf{y}) = \sum_{j=1}^{k} \sum_{i=1}^{n} (u_{ij})^m \lVert \mathbf{x}_i - \mathbf{y}_j \rVert^2 < \varepsilon$ **(6.26)**

Table 4 – FCM Algorithm

6.4.2.2 Selecting the FCM Parameters

As with the k-means algorithm, there are a number of parameters to determine when implementing the FCM algorithm. Many of them are the same or they are related. They include selecting the:

- method of initialisation
- exponent of the partition matrix, *m*

- number of clusters, *k*
- distance function
- stopping criterion

Initialisation of FCM

The initialisation in FCM is typically done by randomly generating the fuzzy partition matrix, **U**. This determines which data points belong in part or in whole to which centres. Other methods could be used which used some pre-existing knowledge of the data structure and would assign the membership values of each data point based on some expert knowledge. This would be very difficult with large data sets and a large number of clusters since the dimensions of the partition matrix are N x k. Due to the nature of the algorithm, it is not possible to assign the values of the centres, since they are calculated from the means and are based on the membership of each data point to the centre.

The Weighting Exponent

A second parameter to consider is the weighting exponent, m, of the partition matrix, **U**. The exponent controls the fuzziness of the membership assignments. The higher the value the more an input vector can be shared between clusters. When m approaches 1 from the positive side, the algorithm approaches hard C-Means, or the K-Means algorithm.

Remaining FCM Parameters

The remaining parameters are selected according to the same considerations as in the K-Means algorithm. Please visit section 4.3.1.2 for information concerning the selection of the parameters dealing with: the number of clusters, the distance function, and the stopping criterion.

6.4.3 Comparing features of the Objective Function Algorithms

Table 5 outlines and compares the primary features of the algorithms:

Feature	K-Means	FCM
Theoretically	Simple	More Complex
Computationally	Simple	More Complex
# of Parameters	5	5
Robustness	Less	More
Cluster Boundaries	Crisp	Fuzzy

Table 5 – Comparison of Three Objective Function Clustering Algorithms

Ultimately, the Table shows that:

- K-Means is theoretically and computationally simple
- FCM has more flexibility is modelling the clusters because it allows fuzzy boundaries between clusters and is computationally more intense. The flexibility produces a more robust result.

The algorithms follow similar processes as shown in Table 6:

	K-Means	FCM
Initialisation	Most often random or expert	Random, expert partition more difficult
Step 2	Assign input vectors to a prototype by calculating the Argmin of distance function	Compute means of fuzzy clusters
Step 3	Compute means	Update partition matrix
Continuation	Stop when $\Delta J<\varepsilon$ or iterations = max	Stop when $\Delta J<\varepsilon$ or iterations = max

Table 6 – Summary of Equations for Three Objective Function Methods

6.5 Implementation of a Clustering Algorithm

The FCM algorithm was selected to generate cluster centres. The resulting cluster centres are then used as prototypes in a classification system that assigns an input vector to one of *k* posture groups. The FCM algorithm was chosen over the K-Means algorithm because of the added flexibility provided by the algorithm.

The algorithm shown in Table 5 was implemented. Parameters of the algorithm were set as follows:

Parameter	Implemented
Method of initialisation	Random
Exponent of the partition matrix, *m*	2
Number of clusters, *k*	20, 25, 30, 35, 40, 45, 50
Distance function	Euclidean Distance
Stopping criterion	Maximum iterations of 500, 100 and 70 or $\varepsilon = 1.0\text{x}10^{-5}$

Table 7 – Implemented Parameters of FCM

Random initialisation was chosen since it is the simplest implementation and because it generally produces sufficient results. The exponent of the partition matrix was set to 2 since this seemed to offer a good compromise between hard-c means and over fuzzification of the memberships. The number of clusters was set to all values shown in Table 7 so that the optimal number of clusters could be extracted. It is not possible to determine this number theoretically. The Euclidean distance function was chosen since it offers both advantages of the Minkowski and the Mahalanobis distances. It was also selected since the data was assumed to form hyper-elliptical clusters. Finally, both stopping criteria were implemented. The threshold, ε, was kept at

1.0×10^{-5} during all runs, but different runs were implemented with the maximum number of iterations equal to 500, 100, and 70. In this mode the algorithm ran until the threshold value was met or until the maximum number of iterations occurred.

The resulting prototypes are used in a classifier, or mapping, system. We compare the input vector to each of the prototypes and then label the vector with the posture group number that it is closest to according to the Euclidean distance measure.

6.6 Results According to Clustering Parameters

While it would be useful to assess, quantitatively and directly, the suitability of cluster prototypes immediately after generating them, it is difficult to do this since we are in the middle of the recognition process. We can still gain useful insight by checking the convergence of the objective function and the distribution of the input vectors with respect to the prototypes.

6.6.1 Convergence of FCM Objective Function

Figure 34 shows that the objective function converges towards a solution and does not oscillate. It also shows that the objective function is smaller, the greater the number of clusters. We expect this, although it is important to note that a smaller value in the objective function does not necessarily guarantee a better solution.

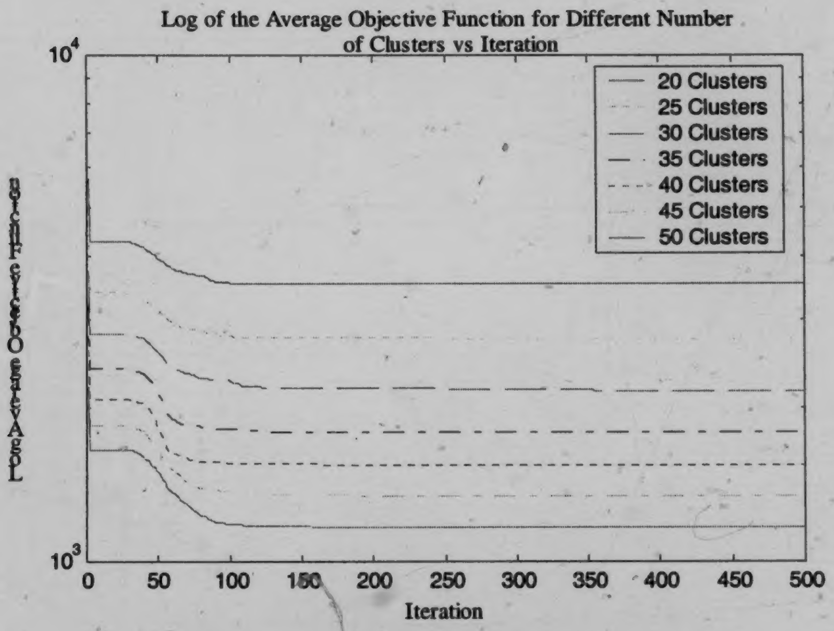


Figure 34 - FCM Objective Function Converging

6.6.2 Sample Prototypes and Distribution of Data

Table 8 shows a sample of prototypes resulting from the FCM algorithm. Note that these prototypes gave the best gesture recognition results when attempting to recognize 8 gestures. Notice, there are a number of prototypes that are appear identical (see the highlighted rows). They actually vary slightly in the higher decimal places.

	Input Sensor							
Prototype	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2	1.01	1.02	2.02	2.96	2.99	2.97	1.02	1.01
3	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
4	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
5	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
6	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
7	1.37	1.53	1.72	1.63	1.66	1.52	1.51	1.40
8	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
9	1.01	1.00	1.00	1.00	1.00	1.01	2.00	2.00
10	1.02	1.00	1.00	1.00	1.00	1.02	2.99	2.99
11	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
12	1.00	1.00	1.00	1.00	2.00	2.00	1.00	1.00
13	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
14	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
15	1.01	1.01	1.03	1.98	2.02	1.99	1.02	1.01
16	2.90	2.99	2.97	2.95	2.99	2.87	2.99	2.95
17	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
18	2.99	2.98	1.01	1.00	1.00	1.01	1.01	1.01
19	2.08	2.98	2.99	2.98	2.99	2.94	1.02	1.02
20	2.99	2.99	2.01	1.01	1.01	1.01	1.01	1.01
21	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
22	1.01	1.02	1.98	2.01	2.97	2.01	1.02	1.01
23	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
24	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
25	1.08	2.93	2.98	2.97	2.96	2.03	1.03	1.02
26	2.96	2.97	2.04	1.99	2.96	2.95	2.97	2.91
27	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
28	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
29	2.96	2.99	2.98	1.98	1.02	1.01	1.01	1.01
30	2.00	2.00	1.01	1.00	1.00	1.00	1.00	1.00
31	1.36	1.41	1.41	1.29	1.34	1.35	1.34	1.27
32	1.01	1.01	1.01	1.00	1.01	1.02	2.99	2.00
33	1.99	2.98	2.00	1.01	1.01	1.01	1.01	1.01
34	1.04	1.99	2.96	2.95	1.96	1.05	1.02	1.02
35	1.00	1.00	1.01	1.02	2.98	2.98	1.01	1.00

Table 8 – Sample of Prototypes Resulting from FCM with k=35

Also, when analysing the data it appears that the input vectors that are mapped to prototypes 3 and 27 typically have one sensor value that is assigned to the 2nd interval in the pre-processing stage while all the others are assigned to the 1st interval. The problem is that the value mapped to the 2nd interval can be from any one of the 8 sensors. This is a problem because it groups the postures of a hand being near any one sensor on the robot into the same categories as a hand being near any one other sensor. A distance function that is better suited to the data, such as one similar to the image similarity measure, would likely solve this problem.

Figure 35 shows the frequency of input vectors mapped to each prototype. When comparing this graph to the table above, it should be noticed that only two of the nearly identical prototypes, numbers 3 and 27, actually end up representing any data. This may be a sign that the exponent of the partition matrix is too large.

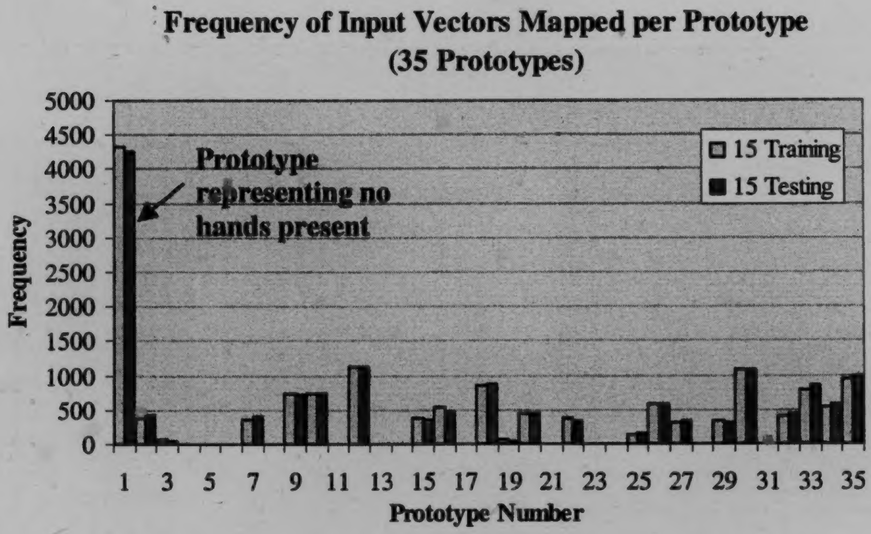


Figure 35 - Distribution Input Vectors with Respect to Prototype

One observation is that the testing data set and the training data set yield very similar distributions.

One should also notice that the first prototype represents a very large number of input vectors. This prototype represents the instance when there are no hands present. Recall that we recorded input data for 13 instances once the presence of a hand triggered the robot to begin sending data. Next, we will explain how we attempted to remove the influence of the dominance of these values.

We used two approaches to find the cluster prototypes. In one method, when feeding the input vectors into the clustering algorithm, we removed all of the instances of input vectors when the robot did not detect any hands. The second method performed clustering on all instances. Tables 9 and 10 show the best recognition results for both methods. In order to show how the values are not always better, we have included values for differing numbers of iterations as well.

Iterations of FCM	Zeros Removed		Zeros Included	
	Training	Testing	Training	Testing
70	83.6	84.1	87.0	**89.8**
100	88.3	90.9	85.3	86.3
500	**90.6**	**93.3**	**89.2**	89.7

Table 9 – The Effect of Removing the Zero Instances When Classifying 8 Gestures

Iterations of FCM	Zeros Removed		Zeros Included	
	Training	Testing	Training	Testing
70	80.2	80.2	**86.8**	**88.2**
100	79.0	81.0	81.4	81.9
500	**81.8**	**81.0**	84.8	86.7

Table 10 – The Effect of Removing the Zero Instances When Classifying 15 Gestures

Tables 9 and 10 also show that when the zeros are excluded from the clustering data set that the classification results improve as the number of iterations of the algorithm is increased. However, when we include the zeros, the relationship between the number of iterations and the classification rate is unknown. This suggests that we may benefit from performing more iterations on a data set with the zeros removed. The stopping criterion of $\varepsilon = 1.0x10^{-5}$ would also need to be decreased since it generally stops algorithm from repeating before 500 iterations are actually complete.

We originally collected data samples of 22 different gestures and used all of this data to form the clusters. The distribution of the input data with regards to the prototypes for the case of all 22 gestures and samples of 15 and 8 different gestures is shown in Figure 36. It shows that there are no prototypes used by all 22 gestures that are not used by either the 15 or 8 other gestures. This indicates that the postures groups are similar for all categories of gestures.

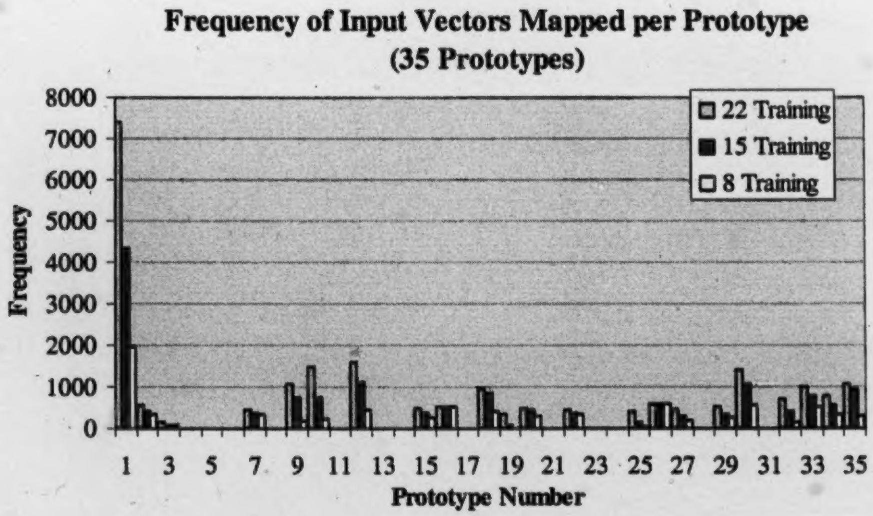


Figure 36- Distribution of Input Vectors by Number of Gestures

6.7 Chapter Summary

We have seen in this chapter that clustering seeks similarities between data samples in a subset. There are a number of different clustering procedures and not one of them is universally applicable. There are three major categories of clustering: hierarchical, graph theoretical, and objective functional. The properties of each of the methods are used to determine which one to apply to a given problem. The desired outcome of the clustering process and the data structure are important factors to consider when selecting a method to use.

We have chosen to implement an objective functional clustering procedure, namely FCM, because it is well suited to the hyper-elliptical nature of our data. It is also suitable because we want to implement a classification system with the prototypes of the clusters and prototypes are easily extracted from the FCM procedure.

The K-means procedure is the simplest of the objective functional methods and so details relating to it were presented prior to explaining the FCM method. We learned how the algorithms work and how to set the parameters.

After comparing K-Means to FCM, we explained how the FCM procedure was implemented in this experiment. We use clustering to find posture groups from the pre-processed input vectors. The prototypes of these posture groups are then used in a classifier, or mapping, system. This enables us to pass only the labels of the posture group onto the HMM, thereby simplifying the computations of the entire system.

The final portion of the chapter presents results of the clustering stage. We see that the objective function of the FCM converges and does not oscillate towards a solution. We have seen a sample of the prototypes used when classifying 8 gestures. It appears that a different distance function may be too better suited to the problem. We have seen that there are no additional posture groups found when we map 15 or 22 gestures. As such, it is better to extract prototypes from clustering 22 gestures since we have a larger data set.

Chapter 7: Hidden Markov Models

In this chapter we present Hidden Markov Models (HMM). We begin by presenting Finite State Machines so that we may compare them to HMM. We then present background information about HMM. We show how they are a more flexible extension of Discrete Markov Processes. We then explain the three basic problems of HMMs and explain their solutions. Following that, we explain how HMMs are implemented in the gesture recognition system of this experiment and give some examples of HMM that represent particular gestures.

7.1 Finite State Machines

Finite state machines (FSM) are temporal models with states. They include a way to describe the initial state and we move from one state to another depending on the current state and current input to the model and the transition function. The output of the machine depends on the output function of the current state. A given sequence of inputs produces a given sequence of outputs. What makes an FSM finite is that it has a fixed number of states.

To summarize, consider a finite state machine to contain:

- a set of input events,
- a set of output events,
- a set of states,
- a function that maps states and input to output,
- a state transition function that maps states and inputs to states, and
- a description of the initial state [38].

There are a number of variations of FSMs that are affected by selecting the functions and the description of the initial state. Discrete Markov Processes and Hidden Markov Models are variants of FSM. The major difference is that they have no input but only output sequences and the functions that describe them relate only to state transition and output functions of each state. They are more suited to this model since we do not have a one to one relationship between inputs and outputs. Essentially, we will consider the collection of postures to be the observation sequence of a gesture and consider like the output of the state machine. We will have one machine to represent each gesture.

7.2 Background

HMM are statistical signal models. They are an extension of discrete Markov processes. Discrete Markov processes are basic state models with probabilities assigned to the state transitions [12]. Each state represents an output and state transitions occur at evenly spaced time intervals. For example, if we had a coin and tossed it every 30 seconds, it could land in one of two states: heads or tails. Given a fair coin, the probability of moving from one state to another would be 0.5 and the probability of remaining in a given state would be 0.5. The resulting discrete Markov process is represented in Figure 37.

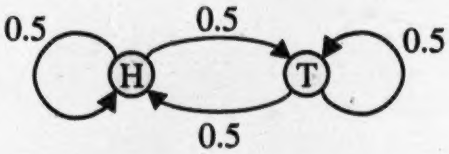
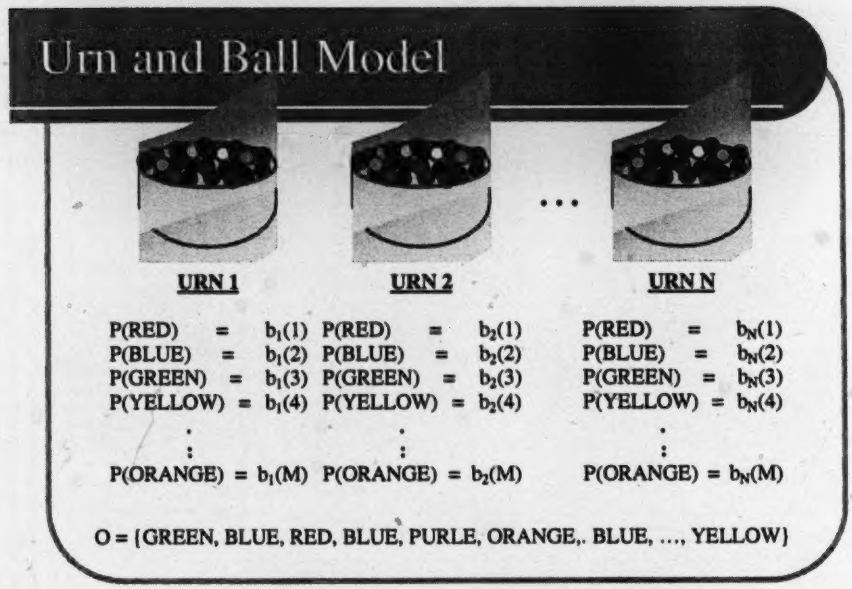


Figure 37 – Discrete Markov Process

HMM add a layer of complexity and flexibility to the model. They allow for more variety in data. Instead of every state representing an output, it represents a set of emission probabilities. For example, say our friend now had two fair coins. He was tossing them one at a time at regular intervals. He would still tell us if the result was heads or tails. In a hidden Markov model, each coin could be assigned a state and the emission probabilities would be 0.5 heads and 0.5 tails in each state. The transition probabilities would be the probabilities associated with changing or not changing coins.

The model is considered hidden, because we only know the outcome and not necessarily which coin was tossed. The increased level of complexity also leads to an increased level of flexibility when trying to model signals or gestures. The flexibility allows for variations in training data.

To further explain HMMs, let us examine a more complicated example, Ferguson's Urn and Ball model [36]. Figure 38 depicts the situation; there are *N* urns, each of which contains a number of balls that can be any of M distinct colours. Now, imagine that there is a genie in the room and he is picking out balls according to some random process. The genie begins by selecting an initial urn. He then picks a ball from the urn, records the colour, and then replaces the ball into its urn. The genie then selects a new urn or returns to the same one. The urn and ball selection and observation process is repeated at regular intervals. The observation sequence which is recorded is considered the observable output of the HMM.



Urn and Ball Model

URN 1

P(RED) = $b_1(1)$
P(BLUE) = $b_1(2)$
P(GREEN) = $b_1(3)$
P(YELLOW) = $b_1(4)$
:
P(ORANGE) = $b_1(M)$

URN 2

P(RED) = $b_2(1)$
P(BLUE) = $b_2(2)$
P(GREEN) = $b_2(3)$
P(YELLOW) = $b_2(4)$
:
P(ORANGE) = $b_2(M)$

…

URN N

P(RED) = $b_N(1)$
P(BLUE) = $b_N(2)$
P(GREEN) = $b_N(3)$
P(YELLOW) = $b_N(4)$
:
P(ORANGE) = $b_N(M)$

O = {GREEN, BLUE, RED, BLUE, PURLE, ORANGE,. BLUE, …, YELLOW}

Figure 38 – Ferguson's Urn and Ball Model [36]

We can expand this analogy to help understand the problems we face in a gesture recognition system. In such a recognition system, we would use one room to represent each gesture that we are trying to model. Each room would have its own genie, set of urns, and a different number of each colour of ball in each of the urns. Let us represent the posture groups that result from the clustering stage as different coloured balls. Once we present the observed sequence of coloured balls, generated from a newly made gesture, to each of the genies, he can tell us the likelihood that that observation sequence could have come from his room. The room that produces the highest probability is selected to represent the gesture. Essentially, we build the rooms and train the genies with a collection of examples of each gesture that we are trying to model. We will further address the problem of how to find rooms with genies in the section on *The Three Problems of HMM*, section 7.3.

Table 11 offers a summary of the elements that make up the HMM.

Element	Meaning
$V = \{v_1, ..., v_M\}$	The codebook of observable output
M	Number of distinct observable output
N	Number of states in a model
$A = \{a_{ij}\}$	Probabilities of state transitions (from state S_i to state S_j)
$B = \{b_j(k)\}$	Probabilities of state emissions (of seeing k in state S_j)
π_i	Probabilities of starting in a given state S_i
$\lambda = \{\pi, A, B\}$	The complete model of a particular HMM
O	Observation Sequence

Table 11 – Parameters of Basic HMM

Let us further examine these elements and see how they relate to the Urn and Ball example:

- $V = \{v_1, ..., v_M\}$ represents what is called the codebook of the HMM. This is, essentially, a collection of all of the possible output of the model. In the case of the Urn and Ball model, V contains the list of all of the colours of balls in the urns. V={red, blue, green, yellow, ..., orange}
- M is the number of distinct observable outputs per state. It corresponds to the number of different colours of balls in each urn.
- **A** is an *N*x*N* matrix, where N is the total number of states, or urns, in the model. Each entry, a_{ij}, in **A** represents the probability of transition from state S_i, to state S_j, or from urn i to urn j:

$$a_{ij} = P[q_t = S_j \mid q_{t-1} = S_i], \quad 1 \le i, j \le N \qquad (7.1)$$

where

$$a_{ij} \ge 0 \qquad (7.2)$$

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad (7.3)$$

- **B** = {$b_j(k)$} represents the probabilities of state emissions. In other words, $b_j(k)$ represents the probability of seeing output or coloured ball k, in state or urn j. In the example the number corresponds to the number of balls of colour k in the urn over the total number of balls in the urn:

$$b_j(k) = P[v_k \text{ at } t \mid q_t = S_j], \quad \begin{array}{l} 1 \le j \le N \\ 1 \le k \le M \end{array} \qquad (7.4)$$

- π_i represents the probability of starting in state S_i. It corresponds to the probability of the genie picking a particular urn i to begin drawing balls from:

$$\pi_i = P[q_1 = S_i], \quad 1 \le i \le N \qquad (7.5)$$

- λ = {π, A, B} represents all of the parameters in the model. In other words, it is a collection of **π**, **A**, and **B**. It is a convenient way to represent the model
- **O** is the sequence of observed output of the HMM or the observation sequence. It corresponds to the final list of colours recorded by the genie each time he removed a ball from the urns up until time T:

$$\mathbf{O} = O_1 O_2 \cdots O_T \qquad (7.6)$$

Let us further consider the example that relates to human gestures performed towards a robot. We will assume that the robot is capable of discerning the posture and position of hands around it and that the postures are sampled at a fixed interval of time. Table 12 shows the relationship between model parameters and how they relate to this example.

Element	What It Represents in Gesture Modelling
V= $\{v_1, ..., v_M\}$	The codebook of possible hand postures
M	Number of hand posture groups
N	The total number of states used to represent each gesture
A = $\{a_{ij}\}$	The probability of moving from one state in the model to another or remaining in the same state
B = $\{b_j(k)\}$	Probabilities of seeing a hand posture from group k in state, S_j
π_i	The probability of starting in a particular state, S_i
$\lambda = \{\pi, \mathbf{A}, \mathbf{B}\}$	The complete model of a particular gesture
O	The observed sequence of hand posture groups

Table 12 – Example of How HMM can Model a Gesture

The codebook, **V**, is comprised of all of the possible hand postures, **M**, that can be made when performing gestures. The observation sequence is the collection of hand postures sampled at regular intervals by the robot sensors. Unfortunately, there is no concrete way to represent how many states there are, N, what the states are or what the probability of moving from state S_i to state S_j, a_{ij}, is. We also cannot directly represent the probability of starting in a given state, π_i. This is because, in this example, the state of the model is hidden from observation and we only know what is observed, **O**. Given this fact, we cannot know which state we are in and which state we move to. We can, however, represent the probabilities of state emissions, $b_j(k)$, as the probability of seeing a hand posture, k, in state S_j.

We are trying to build a hidden Markov model of each gesture for recognition. Unfortunately, there is no genie in the robot so we cannot ask him what the parameters, **λ**, of the models are. The only data that we have are the sequences of training data captured from the robot sensors. Fortunately, the sequences are labelled with their gesture name. This is typical of the problems that use HMM to model words, signs, letters or any other sequence of data in time.

7.2.1 HMM Structure

Even before we attempt to determine the parameters, **λ**, of the model, we need to define the structure of the HMM. This includes determining the underlying connections within the models as well as values for M, $\mathbf{V}$ and finally the number of states N. The format of the observation sequence has the biggest impact on the selection of structure for the HMMs. The observation sequence, **O**, can also be either finite in length, have obvious beginning and end points, or be a segment of an infinite sequence. Instances of the observation sequence, **O**, could be comprised of either a discrete point or a multivariate vector.

7.2.1.1 Ergodic versus Left to Right

HMMs are easily divided into three main categories of structure:

- fully connected or ergodic,
- left to right, or
- any other variation.

In an ergodic model the state transition probabilities are always greater than 0, $a_{ij} > 0 \; \forall \; i,j$. In other words, we can reach any state from any state in the model. We can also continue to travel in the graph indefinitely. See Figure 39 for and example of a 4 state ergodic model.

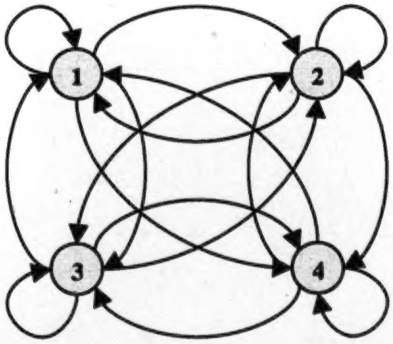


Figure 39 – An Example of a 4 state Ergodic Model [36]

In left to right models we can only travel to states that are in the same place or further to the right of the current state. That means that we always start on the left in state 1 and that the probability of moving from any state where *j* is less than *i* is zero; $\pi_1 = 1$ and $\pi_i = 0 \ \forall \ i > 1$ and $a_{ij} = 0 \ \forall \ i < j$. There is often a limit placed on the permissible length of state jumps, say for example $a_{ij} = 0 \ \forall \ |j - i| > 2$. This is done to prevent ending up in the final state of a model early on in an observation sequence. The observation sequence should be of finite length or we will eventually be trapped in the final state of the model. Figure 40 provides an example of a 4 state left to right model with the state jump limited to 2.

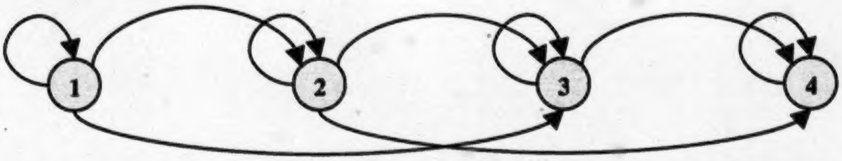


Figure 40 – An Example of a Left to Right Model with the State Jump Limited to 2 [36]

The final category includes any other variation of model that is not defined by the preceding rules. An examples of such a model is shown in Figure 41.

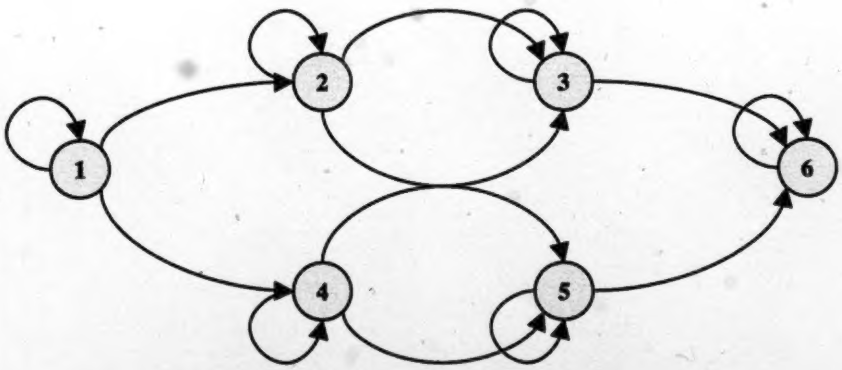


Figure 41 – An Example of a 6 State Parallel Left to Right Model [36]

Ergodic models lend themselves well to the case of infinite, or segments of infinite, observation sequences. Left to right models are best suited to cases where there is a clear beginning and end to the observation sequence. This is the case with spoken words and most gestures. The third category can be suited to either case depending entirely on the structure.

7.2.1.2 Multivariate versus Discrete Observations

We will now consider values for *M* and **V**. These values are, once again, entirely dependent upon the observation sequence and are obviously related. M is simply the number of codes in **V**. Up until this point, it has been assumed that the observation sequence is comprised of only discrete points which can be used to formulate the codebook **V**. In the case of the urn and ball model, **V** was simply comprised of all of the colours of balls selected by the genie. Data is rarely presented in such a convenient manner. Most often, each instance of the sequence presented takes the form of a multivariate vector.

This leaves the modeller with one of two choices: perform data quantisation by extracting features from the vectors and then assigning each instance to a value from the codebook, or determine the model parameters, **λ**, according to the multivariate vectors. The second option has the advantage of losing the spatial-temporal relationship of the vectors before creating the HMM, but has the disadvantages of being more computationally expensive and theoretically complicated. In either case, the modeller is still left with a number of decisions to make.

Data Quantized Observation Sequence

Let us begin by examining the case where features are extracted from the multivariate vectors and data quantisation is performed by assigning the instances to a value from the codebook. One needs to decide:

- which features to extract,
- how to map instances to the values of the codebook, and
- the size and composition of the codebook.

Feature extraction is an important step in most machine learning problems. Data often includes a great deal of irrelevant or less relevant information. By properly extracting only the most pertinent features of the data, computations are greatly simplified and the results are greatly improved. Features may include angles between subsequent data points, velocity, rotation, density, intensity, colour, size, and many others. Features are extracted via pre-processing of the data.

Extracted features can be used to find clusters in the data. In the case of HMMs with discrete observation sequences, the prototypes of the clusters are used to represent the labels in the codebook. Each instance of the multivariate observation sequence can be mapped to the prototype that best represents it. The prototype labels, in turn, form the discrete observation sequence. The K-Means clustering algorithm is most frequently used in the literature, but other methods are also applicable. The size and composition of the codebook is now dependant on the clusters that best represent the features of the data. Thus, M and V are selected via the clustering procedure.

Using a Multivariate Observation Sequence

In the case of preserving the multivariate vector, one needs to determine what pre-processing to perform and the nature of the probability density function, $b_j(\mathbf{O}_t)$, that represents the emissions probabilities in each state S_j. It needs to be represented in a form that allows parameters to be re-estimated; this is in accordance with the learning method proposed as the solution to problem 3 below. The typical form of the densities is:

$$b_j(\mathbf{O}) = \sum_{m=1}^{M} c_{jm}\mathcal{N}[\mathbf{O},\boldsymbol{\mu}_{jm},\mathbf{U}_{jm}], \quad 1 \le j \le N \qquad (7.7)$$

$b_j(\mathbf{O})$ represents a finite mixture of densities where:

- $\mathbf{O}$ is the vector being modelled
- c_{jm} represents the mixture coefficient for the m^{th} mixture in state S_j,
- $\mathcal{N}$ represents any log-concave or elliptically symmetrical density function
    - Typically a Gaussian density function is used,
- $\boldsymbol{\mu}_{jm}$ represents the mean vector of the density function, and
- $\mathbf{U}_{jm}$ represents the covariance matrix.

In this format, one must determine the density function, $\mathbf{N}$, to model and the value of m, in other words how many density functions to mix.

7.2.1.3 Number of States

This brings us to the final element in the structure of an HMM, how many states should there be in the model? There is no easy answer to this question and there are no quantifiable methods that exist to help determine this number. A couple of suggestions have been made with regards selecting the number of states in models used to represent words in speech recognition. The first being to have as many states as there are sounds, or phonemes, in a word The second being to have the number of states correspond to the number of observations in the word.

One needs to review existing implementations of HMMs to gain an understanding of model size as applicable to the current problem and would benefit from testing out a few different values for N.

7.3 Three Basic Problems in HMM

We now have an understanding of how to determine the underlying structure of an HMM. What is next? We are now faced with the three basic problems of HMMs:

1) How do we determine the probability of an observation sequence given a model, $P(\mathbf{O}|\lambda)$?
2) Given an observation sequence, **O**, and a model, **λ**, how do we select a state sequence, $\mathbf{Q} = q_1, q_2, \ldots q_T$, that best explains the observation sequence?
3) How do we find the model parameters, **λ**, that maximise the probability of the given observation sequence? In other words, how can we train an HMM from a data set?

7.3.1 Problem 1

The first problem considered is how to determine $P(\mathbf{O} \mid \lambda)$, that is the probability of an observation sequence, **O**, of length, T, given a model, **λ**. The solution to this problem allows the models to be used in a gesture classifier system. When an observation sequence is presented, it can be tested against all available models. The model that yields the greatest $P(\mathbf{O} \mid \lambda)$ is selected as the winner and the gesture is classified as being the one represented by that model.

While $P(O \mid \lambda)$ can be determined explicitly, the number of calculations required when using discrete, ergodic HMMs is $2T \cdot N^T$. Given an observation sequence of length $T = 13$ and with $N = 10$ states, 2.6×10^{14} calculations are required just to test one model. In a classification system there are sometimes more than 20 models, this means that more than 5.2x10^{15} could be required. This could pose a problem in real time recognition systems.

A solution that is easier to implement is the forward-backward procedure [36]. It is a procedure that recursively determines forward parameters, α, and backward parameters, β. Using this procedure, $P(\mathbf{O} \mid \lambda)$ can be found in N^2T calculations in discrete ergodic HMMs. The variables, α and β, are also used in solving problem three.

7.3.1.1 Forward Procedure

Let us begin with the forward variable, $a_t(i)$. It represents the probability of the partial observation sequence, $O_1\, O_2 \cdots O_t$ and state $S_i = q_t$ at time t given the model λ. It is defined as:

$$\alpha_t(i) = P(O_1\, O_2 \cdots O_t, q_t = S_i \mid \lambda). \qquad (7.8)$$

The forward variable is solved inductively as shown in Table 13 [36].

Forward Procedure		
Initialisation	$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N$	(7.9)
Induction	$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \le t \le T-1$ $1 \le j \le N$	(7.10)
Termination	$P(\mathbf{O} \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i).$	(7.11)

Table 13 – The Forward Procedure

The initialisation step sets $a_1(i)$ for all i, 1<= i <= N as the joint probability of seeing the initial observation O_1 in state S_i. Induction is used to find $a_{t+1}(i)$.

This step begins by computing the term $\alpha_t(i)a_{ij}$. Essentially it calculates the joint probability of seeing $\mathbf{O} = O_1\, O_2 \cdots O_t$ and reaching state S_j from S_i at time $t+1$. This calculation is repeated for all N states S_i and then summed for all i. The resulting sum is the probability of being in state S_j at time $t+1$. Finally, we can find the joint probability of being in state Sj at time $t+1$ and seeing O_{t+1}. Termination occurs once $\alpha_T(i)$ has been found. $\alpha_T(i)$ represents the probability of seeing the complete observation sequence, $\mathbf{O}$ and being in state S_i, $P(\mathbf{O}, q_T = S_i \mid \lambda)$. Finally the probability of the observation sequence given the model, $P(\mathbf{O} \mid \lambda)$ can be found by summing $\alpha_T(i)$ for all N states.

7.3.1.2 Backward Procedure

As we can see, we have already found a solution to problem 1. Regardless of this, we will continue to present the backward procedure. There are two reasons for this: the backward procedure can also be used to solve problem one and it will be required to solve both problems 2 and 3. The procedure follows the same principles as the forward procedure the difference is that it starts at the end, or back, of the observation sequence instead of the beginning, or front.

$\beta_t(i)$ represents the probability of the partial observation sequence, $O_1\, O_2 \ldots O_t$ given that we are in state $S_i = q_t$ at time t and given the model λ. It is defined as:

$$\beta_t(i) = P(O_{t+1}\, O_{t+2} \cdots O_T, q_t = S_i \mid \lambda). \qquad (7.12)$$

The backward variable is solved inductively as shown in Table 14 [36].

Backward Procedure		
Initialisation	$\beta_T(i)=1,\quad 1\le i\le N$	(7.13)
Induction	$\beta_t(i)=\sum_{j=1}^{N} a_{ij}b_j(O_{t+1})\beta_{t+1}(j),\quad t=T-1,T-2,\cdots,1$ $1\le j\le N.$	(7.14)

Table 14 – The Backward Procedure

The initialisation step arbitrarily sets $\beta_T(i)$ to 1 for all i. The induction step accounts for being in state S_i at time t by summing the transition probability from state S_j, a_{ij}, the observation of O_{t+1} in state S_j, $b_j(O_{t+1})$, and the remaining partial observation sequence from state S_j, $\beta_{t+1}(j)$.

7.3.2 Problem 2

The second problem involves trying to determine the optimal state sequence given **O** and **λ**. This problem does not need to be solved in order to train models or to classify sequences; although, solving it gives one a better understanding of the models. It is used to analyse resulting models.

There are a number of different ways to approach this problem; they are dependant on what is considered optimal. The most straightforward approach might be to simply choose the state at time t that has the highest probability of producing the current observation, O_t, and to repeat this process until all observations are accounted for. The result would be the most optimal state at each instant in time.

The problem with this approach is it could result in impossible state sequences. That is because subsequent states in the sequence could have a transition probability of zero, $a_{ij} = 0$. The most commonly used criterion is to find the single state sequence, **Q**, that maximises the probability of the state sequence given the observation sequence and the model, $P(\mathbf{Q}|\mathbf{O},\lambda)$. Viterbi algorithm [36] is used to solve this problem.

Let us begin by defining the highest probability along a single path, at time t, that considers the observations up to t and ends in state S_i:

$$\delta_t(i) = \max_{q_1 q_2 \cdots q_{t-1}} P[q_1\, q_2 \cdots q_t = i, O_1\, O_2 \cdots O_t \,|\, \lambda) \qquad (7.15)$$

Using induction, we define:

$$\delta_{t+1}(j) = \max_i\, [\delta_t(i) a_{ij}] b_j(O_{t+1}). \qquad (7.16)$$

Finally, we need to be able to keep track of the argument that maximised this probability for both t and j. We will track this using the $\psi_t(j)$ array.

Viterbi Algorithm		
Initialisation	$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N$	(7.17)
	$\psi_1(i) = 0,$	(7.18)
Recursion	$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \quad 2 \le t \le T$ $1 \le j \le N$	(7.19)
	$\psi_t(j) = \arg\max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], \quad 2 \le t \le T$ $1 \le j \le N$	(7.20)
Termination	$P^* = \max_{1 \le i \le N} [\delta_T(i)]$	(7.21)
	$q_T^* = \arg\max_{1 \le i \le N} [\delta_T(i)]$	(7.22)
Path (state sequence) backtracking	$q_t^* = \psi_{t+1}\, [q_{t+1}^*], \quad t = T-1, T = 2, \cdots, 1.$	(7.23)

Table 15 – The Viterbi Algorithm [36]

The initialisation step assigns the joint probability of being in state S_i and seeing the first observation O_t. We also set the value of $\psi_1(i)$ to zero. This is repeated for all states N. The first part of the recursion step is similar to the induction used in the forward procedure; the difference is that $\delta_{t-1}(i)a_{ij}$ is maximised instead of being summed. In addition the argument that maximised in equation 7.17 is tracked in the $\psi_t(j)$ array. Termination also follows the same methods as with the forward procedure. Once the final state is found, path backtracking is used to find the "optimal" state sequence.

7.3.3 Problem 3

We now return to the problem presented above: given that we have observation sequences $\mathbf{O}$, how can we find the model parameters, $\boldsymbol{\lambda}$? This is a very important problem since we cannot solve problem 1 or 2 unless we have a model to work with. It is also the most difficult as it cannot be solved analytically.

The solution to problem 3 involves optimising the model parameters $\mathbf{A}$, $\mathbf{B}$, and $\boldsymbol{\pi}$. There are two main methods used to solve this problem; the first is gradient descent and the second is the Baum-Welch method (also known as expectation maximisation or EM) [36]. EM is the typical solution to the problem therefore we will examine this case.

The process iteratively optimises the model parameters. The actual formulae used for updating the parameters in the EM method depend on the structure of the HMM as well as the nature of the observation sequence. It will vary depending on whether the models are ergodic or left to right and whether the observation sequences are discrete data points or multivariate vectors.

7.3.3.1 Parameter Re-estimation with Discrete Valued Observation Sequences

The basic presentation of the EM method for HMM deals with discrete valued observation sequences; it also deals with a fully connected, ergodic, models.

Ergodic Models

Before we present the algorithm, we define ξ as the probability of being in state S_i at time t and state S_j at time $t+1$ given the observation sequence and the model.

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid \mathbf{O}, \boldsymbol{\lambda})$$

$$= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O} \mid \boldsymbol{\lambda})} \tag{7.24}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}$$

We also define γ as the probability of being in state S_i at time t given the observation sequence and the model.

$$\gamma_t(i) = P(q_t = S_i \mid \mathbf{O}, \boldsymbol{\lambda})$$

$$= \frac{\alpha_t(i) \beta_{t+1}(i)}{P(\mathbf{O} \mid \boldsymbol{\lambda})} \tag{7.25}$$

$$= \frac{\alpha_t(i) \beta_{t+1}(i)}{\sum_{i=1}^{N} \alpha_t(i) \beta_{t+1}(i)}$$

We can represent γ in terms of ξ by summing over the terms of j:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j). \tag{7.26}$$

If we sum γ *and* ξ over time from time $t = 1$ to $t = T\text{-}1$, we get the expected number of transitions from state S_i and the expected number of transitions from state S_i to S_j respectively:

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from } S_i \qquad (7.27)$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from } S_i \text{ to } S_j. \quad (7.28)$$

Using these definitions, we can now represent how to re-estimate the model parameters, **π**, **A**, and **B** as follows:

$$\begin{aligned} \bar{\pi}_i &= \text{expected number of times in state Si at time } (t = 1) \\ &= \gamma_1(i) \end{aligned} \qquad (7.29)$$

$$\begin{aligned} \bar{a}_{ij} &= \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i} \\ &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \end{aligned} \qquad (7.30)$$

$$\begin{aligned} \bar{b}_j(k) &= \frac{\text{expected number of times of observing symbol } v_k \text{ while in state } j}{\text{expected number of times in state } j} \\ &= \frac{\sum\limits_{\substack{t=1 \\ \text{s.t. } O_t = v_k}}^{T} \gamma_t(i)}{\sum\limits_{t=1}^{T} \gamma_t(i)} \end{aligned} \qquad (7.31)$$

The current values of the model parameters are used on right hand side of the formulas and they provide updated parameters on the left hand side that more optimally represent the model. Each time the parameters are re-estimated they approach a locally optimal value and in time they converge to a critical point and the iterations stop. Proof of the ability of these formulae to reach local optimums can be found in [36].

Left to Right - Multiple Observation Sequences

As mentioned in section 7.2.1.1, left to right models place certain constraints on the values of **A**. There is also one important consideration to make concerning training the models. In the previous section, only a single observation sequence is used in the parameter re-estimations. Unfortunately, in the case of left to right models, the observation sequences are relatively short and only represent a single occurrence of the gesture. As such, we need to concatenate a number of observation sequences together in order to successfully estimate model parameters:

$$\mathbf{O} = [\mathbf{O}^{(1)}, \mathbf{O}^{(2)}, \cdots, \mathbf{O}^{(k)}] \qquad (7.32)$$

It follows that:

$$\begin{aligned} P(\mathbf{O} \mid \lambda) &= \prod_{k=1}^{K} P(\mathbf{O}^{(k)} \mid \lambda) \\ &= \prod_{k=1}^{K} P_k \end{aligned} \qquad (7.33)$$

We cannot simply utilise these concatenated sequences with the re-estimation equations 7.29-7.32. We need to develop a method of re-estimation that considers that each of the individual sequences, $\mathbf{O}^{(k)}$, must begin in state S_1. The result is that the re-estimation formulae add the frequencies of occurrence for each individual sequence together as follows:

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j(\mathbf{O}_{t+1}^{(k)}) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(j)} \qquad (7.34)$$

and

$$\overline{b}_j(l) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{\substack{t=1 \\ \text{s.t. } O_t = v_l}}^{T_k - 1} \alpha_t^k(i) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k - 1} \alpha_t^k(i) \beta_t^k(j)} \qquad (7.35)$$

Note that there is no calculation required for π_i since $\pi_1 = 1$ and $\pi_i = 0 \; \forall \; i \neq 1$ in left to right models.

Parameter Re-estimation with Multivariate Vectors

We will now consider the case where the instances of the observation sequences are multivariate vectors. The general case of HMM considers the observation sequence to contain no known beginning or end. As such, we will present the ergodic case when examining the multivariate implementation of HMMs.

The major difference between multivariate vector based models and discrete value based models is in the **B** parameter. $b_j(\mathbf{O})$ was previously defined in equation 6.7 above, but we will present it again now:

$$b_j(\mathbf{O}) = \sum_{m=1}^{M} c_{jm} \mathcal{N}[\mathbf{O}, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm}], \quad 1 \leq j \leq N \qquad (6.7)$$

Before presenting the re-estimation formulae for the parameters of **B**, let us represent the probability of being in state S_i at time t and O_t being represented by the kth mixture component:

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \right] \left[\frac{c_{jk} \mathcal{N}[\mathbf{O}, \boldsymbol{\mu}_{jk}, \mathbf{U}_{jk}]}{\sum_{m=1}^{M} c_{jm} \mathcal{N}[\mathbf{O}, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm}]} \right] \qquad (7.36)$$

[22, 29, 36, 40] show that the re-estimation formulae for the elements of **B** follow:

$$\overline{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t(j,k)} \qquad (7.37)$$

$$\overline{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot \mathbf{O}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \qquad (7.38)$$

$$\overline{\mathbf{U}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (\mathbf{O}_t - \boldsymbol{\mu}_{jk})(\mathbf{O}_t - \boldsymbol{\mu}_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)} \qquad (7.39)$$

Since there is no difference in the definitions of a_{ij},or π_i, there is also no change in the method for re-estimating them.

The biggest disadvantage of using multivariate observation sequences is the number of calculations. Let us examine the case where b is modelled with a single Gaussian function [40]:

$$b_j(O_t) = \frac{1}{\sqrt{(2\pi)^n \,|\boldsymbol{\sigma}_j|}} \quad e^{\frac{1}{2}(O_t - \boldsymbol{\mu}_j)'\boldsymbol{\sigma}_j^{-1}(O_t - \boldsymbol{\mu}_j)} \qquad (7.40)$$

where the covariance matrix, $\mathbf{U}_{jk}$, reduces to $\boldsymbol{\sigma}_j$ since there is only one mixture.

7.4 Implementation of HMM in a Recognition System

Our goal in this problem is to create a model that represents each gesture. HMM based gesture recognition systems typically have one model to represent each gesture. These models must be learned from our training data. Once they are trained, testing data or real time sequences are compared to each model using the forward procedure to determine the probability of the observation sequence given the model, $P(\mathbf{O}|\lambda)$. The model that yields that highest probability is selected as the one that represents the gesture made. Figure 42 presents a complete recognition system where each model is shown with N=3. It shows 15 models, λ, one for each gesture in Table 1.

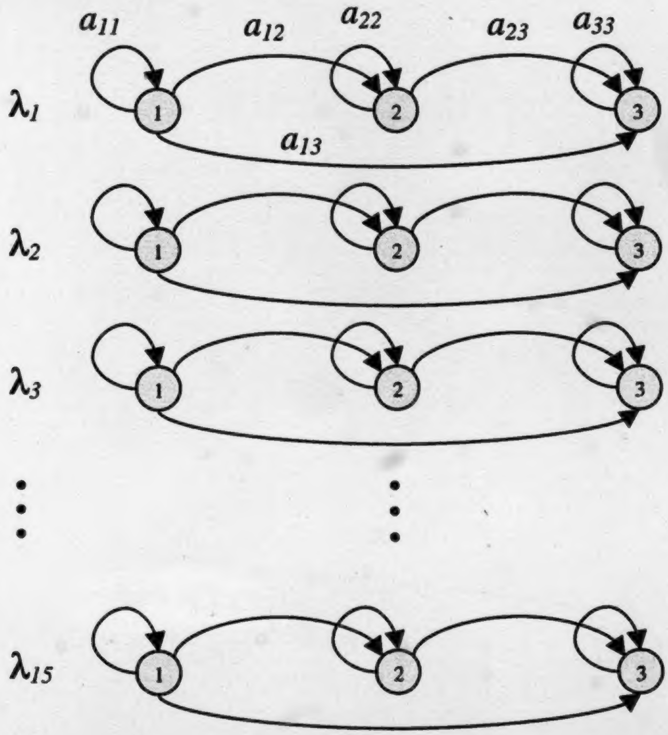


Figure 42 – Collection of Models that Represent the HMMs in the Recognition System

In order to have a successful recognition system a great deal of consideration must be given to the actual implementation of the models. Given all of the information presented in the preceding sections of this chapter, one must decide the best way to implement the HMM given the data. Decisions must be made regarding the structure of the HMM, ergodic models versus left to right or some other variation. One must also decide if the observation sequences presented to models should be the raw multivariate vectors or if they would be better presented as discrete values. The discrete values would result from assigning labels to the multivariate vectors; the labels come from a predetermined codebook. This process requires clustering the data and using the cluster prototypes to classify the sequence vectors at each instance in time.

Left to right models are selected to represent each of the gestures. This is because all of the gestures have a distinct beginning and end. The state jump is limited to 2 in all cases.

As for the number of states per model, the literature [20, 36] was referenced to give an idea for an approximate number of states to use. Because it is not possible to analytically determine the optimal number of states to use, models were built with *2*, *3*, *5*, *10* states.

The next decision relates to how to present the observation sequence to the models, in both training and recognition phases. Serious consideration was given to directly using multivariate vectors as the instances in the observation sequences. While they have the advantage of not losing information by extracting spatial features first, they are both computationally expensive and theoretically complicated.

In order to calculate $P(\mathbf{O}|\lambda)$ using the forward procedure, we need to evaluate $b_j(O_t)$, equation 6.40, $T\mathrm{x}N$ times. Each evaluation of the Gaussian function requires finding the inverse and the determinant of the covariance matrix, σ_j, of dimensionality $d\mathrm{x}d$, where d is the dimensionality of the observation vector. Given that we may have up to 15 models to compare in a recognition system, we would need to evaluate $b_j(O_t)$ up to $15\mathrm{x}T\mathrm{x}N$. The result is that too many calculations are required for real time classification. This is especially true if the implementation of the recognition system was on an embedded system with a slower microprocessor than a full sized computer. For real time implementation in a robotics system, computationally simpler methods are preferred.

As such, clustering is used to perform data quantification of the multivariate vectors. Clustering introduces another step, and a great number of calculations before the HMMs can even be used. The advantage is that, once the cluster prototypes are found, the only calculations required are ones to perform similarity matching of the data instances to the prototypes. In the case of a gesture recognition system, these prototypes represent posture groups. This is significantly less than the *15*x*T*x*N* determinant and inverse matrix calculations used in the multivariate implementation.

7.5 Examples of Trained HMM

We present an example of a 3 state HMM, with 25 posture groups, that represents the gesture "go left", in Table 16. (Recall that we actually tested models with $N = 2, 3, 5,$ and 10 states and all of these models with 20, 25, 30, 35, 40, 45, and $M = 50$ posture groups) The sum of the rows of A should add up to 1 and the dimension is NxN. The sum of the columns of B^T should also add to one and the dimension should be NxM. The vector π should add to 1 and have the dimension N.

Matrix A shows that the probability of staying in state S_1 is 74%, the probability of moving from state S_1 to S_2 is 21% and the probability of moving from state S_1 to state S_3 is 5%.

Recall that the matrix B represents the probability of seeing a posture from each posture group in each state. Not all of the posture groups occur in each model; therefore, many of the entries are zero.

The vector π represents the initial state probabilities. For left to right models the first entry should be 1 and the remaining 0.

We also present a model that models the "go right" gesture, for the case of $M = 35$, $N = 3$, see Table 17. Note that posture groups are not the same, in other words, posture group 5 in the first model does not necessarily represent the same posture group as group 5 in the other model.

The final model, shown in Table 18, is of the gesture "hit right". It is for the case where $M = 50$ and $N = 5$.

$M=$	$N=$	$A=$	$B^T=$	$\pi=$
25	3	$\begin{bmatrix} 0.74 & 0.21 & 0.05 \\ 0.00 & 0.78 & 0.22 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$	$\begin{bmatrix} 0.12 & 0.85 & 0.89 \\ 0.00 & 0.00 & 0.00 \\ 0.03 & 0.06 & 0.02 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.01 & 0.01 & 0.00 \\ 0.21 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.35 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.02 & 0.00 & 0.00 \\ 0.16 & 0.03 & 0.01 \\ 0.00 & 0.00 & 0.00 \\ 0.10 & 0.04 & 0.08 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.01 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix}$	$\begin{bmatrix} 1.00 & 0.00 & 0.00 \end{bmatrix}$

Table 16 – Example of an HMM that Models the Go Left Gesture

$M=$	$N=$	$\mathbf{A}=$	$\mathbf{B}^T=$	$\pi=$
35	3	$\begin{bmatrix} 0.79 & 0.20 & 0.10 \\ 0.00 & 0.90 & 0.10 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$	$\begin{bmatrix} 0.27 & 0.72 & 0.96 \\ 0.00 & 0.01 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.01 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.29 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.14 & 0.03 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.06 & 0.03 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.05 & 0.06 & 0.04 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 \\ 0.01 & 0.02 & 0.00 \\ 0.06 & 0.11 & 0.00 \\ 0.11 & 0.03 & 0.00 \end{bmatrix}$	$\begin{bmatrix} 1.00 & 0.00 & 0.00 \end{bmatrix}$

Table 17 – Example of an HMM that Models the "Go Right" Gesture

$M=$	$N=$	$A=$	$B^T=$	$\pi=$
50	5	$\begin{bmatrix} 0.69 & 0.28 & 0.03 & 0.00 & 0.00 \\ 0.00 & 0.68 & 0.31 & 0.01 & 0.00 \\ 0.00 & 0.00 & 0.08 & 0.66 & 0.26 \\ 0.00 & 0.00 & 0.00 & 0.47 & 0.53 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$	$\begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.29 & 0.40 & 0.10 & 0.21 & 0.18 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.07 & 0.16 & 0.50 & 0.53 & 0.43 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.01 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.07 & 0.10 & 0.08 & 0.05 & 0.02 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.01 & 0.01 & 0.01 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.02 & 0.00 & 0.00 & 0.08 & 0.01 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.09 & 0.09 & 0.00 & 0.07 & 0.05 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.28 & 0.14 & 0.25 & 0.03 & 0.08 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.01 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.16 & 0.10 & 0.06 & 0.02 & 0.22 \end{bmatrix}$	$\begin{bmatrix} 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix}$

Table 18 – Example of an HMM that Models the "Hit Right" Gesture

7.6 Chapter Summary

We presented the background of HMM and showed how they are an extension of Discrete Markov Processes. We chose to implement HMM over Discrete Markov Processes because they offer more flexibility in the model. This is a desirable feature since there is variation in the data. Different people gesticulate at different rates and position their hands in slightly different ways.

We presented the three basic problems of HMM:

1) How do we determine the probability of an observation sequence given a model, $P(\mathbf{O}|\boldsymbol{\lambda})$?
2) Given an observation sequence, $\mathbf{O}$, and a model, $\boldsymbol{\lambda}$, how do we select a state sequence, $\mathbf{Q} = q_1, q_2, \ldots q_T$, that best explains the observation sequence?
3) How do we find the model parameters, $\boldsymbol{\lambda}$, that maximise the probability of the given observation sequence? In other words, how can we train an HMM from a data set?

Subsequently, we presented the solutions to these problems, namely the forward and backward procedures, the Viterbi algorithm, and EM.

There are different structures of HMM and we can see that left to right models best suit our data, since there is a clear beginning and ending to each of the gesture samples. We chose to implement HMMs with discrete observation sequences because they are computationally and theoretically simpler than ones with multivariate observation sequences are. This means that the HMM is only a temporal model of the changes in hand postures and not a complete spatio-temporal model of the gesture. We saw that it is not theoretically possible to determine the number of states in an HMM, as such we generate models with 2, 3, 5, and 10 states. We based these numbers on previous examples in the literature.

Chapter 8: Results of Gesture Recognition

We will now present the results of the gesture recognition process. We will begin by presenting overall gesture recognition results. We will then examine the outcome of the clustering stage and then examine the outcome after the HMM stage.

8.1 Overall Results

We will define the classification rate as the number of correctly identified gestures divided by the total number of gestures that we attempted to identify. We obtained classification rates of 98.1% for the training set and 95.9% for the testing set when attempting to classify between "Caress" and "Direct" categories of gestures. These rates reduce to 93.3% for the training data set and 90.6% for the testing data set when attempting to classify between the 8 individual gestures in these categories.

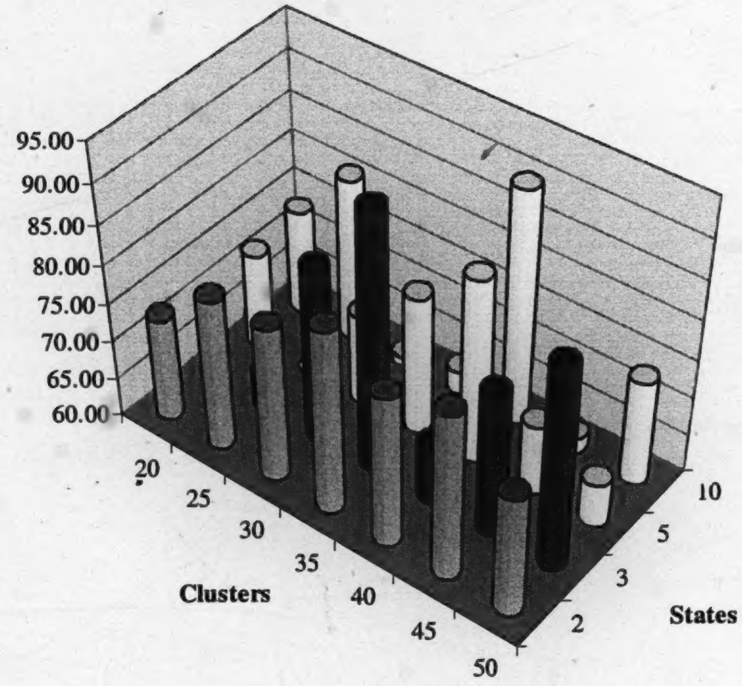
We obtained classification rate of 90.3% for the training data set and 89.3% for the testing data set when attempting to classify between all major categories of gestures ("Caress", "Play", "Attack", and "Direct"). These rates reduce to 88.7% for the training data set and 86.7% for the testing data set when attempting to classify all 15 gestures.

8.2 Results According to the Complete Recognition System

Let us examine the relationship between the number of clusters (which is also the size of the codebook), the number of states in an HMM, and the classification rates. We will present two examples, first for 8 gestures recognized and then for 15.

Let us begin with 8 gestures, Figure 43 (a) and (b). For the sake of comparison, we will fix the maximum number of iterations of the FCM algorithm to 500 and will exclude the input vectors that signify there is no hand present from the clustering stage and set the number of intervals in the pre-processing stage to 3. These values were selected since they produce the best result overall results when classifying 8 gestures when there are 35 cluster prototypes and 3 states.

Classification Rate of Training Data



(a)

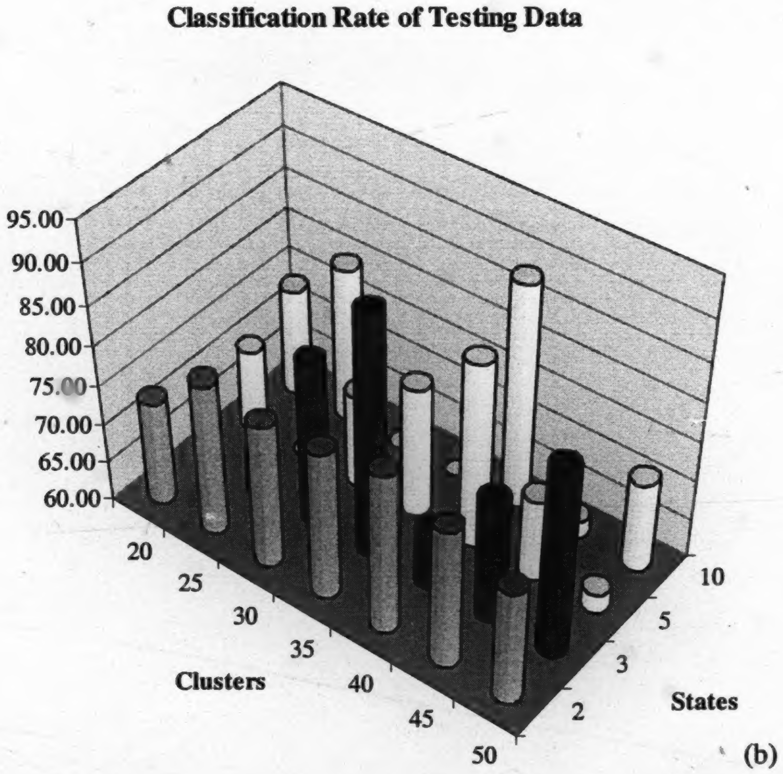
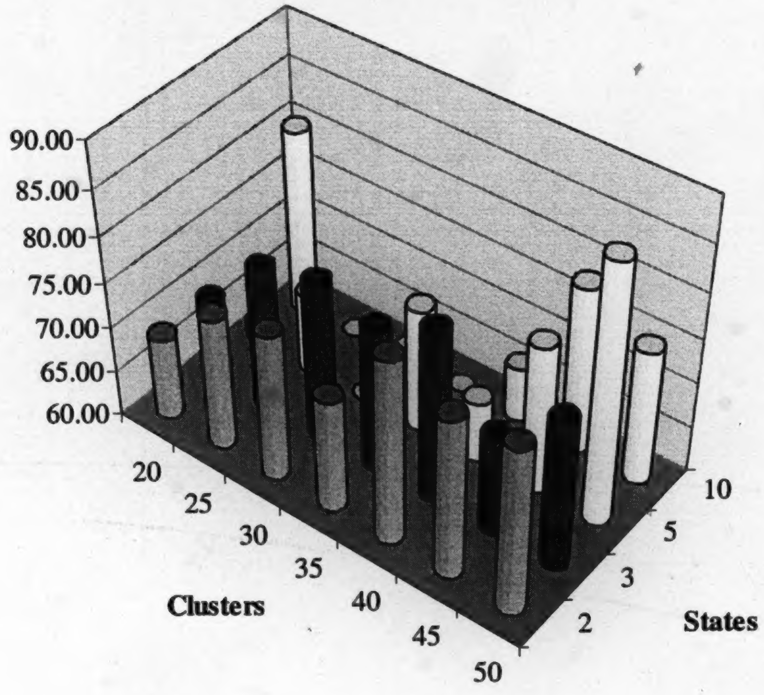


Figure 43 – Classification Rate According to Number of States and the Number of Prototypes for Training (a) and Testing Data (b) with 8 Gestures Recognized

One can see that the solution space is multimodal. This is not surprising since both the FCM algorithm and the HMM algorithm find only the local optimum and not global optimum.

Now consider the 15 gestures classification system, Figures 44 (a) and (b). Where the maximum number of iterations is fixed at 500, zeros are included and set the number of intervals in the pre-processing stage to 4. These values were selected since they produce the best result overall results when classifying 15 gestures when there are 50 cluster prototypes and 5 states.

Classification Rate of Training Data



(a)

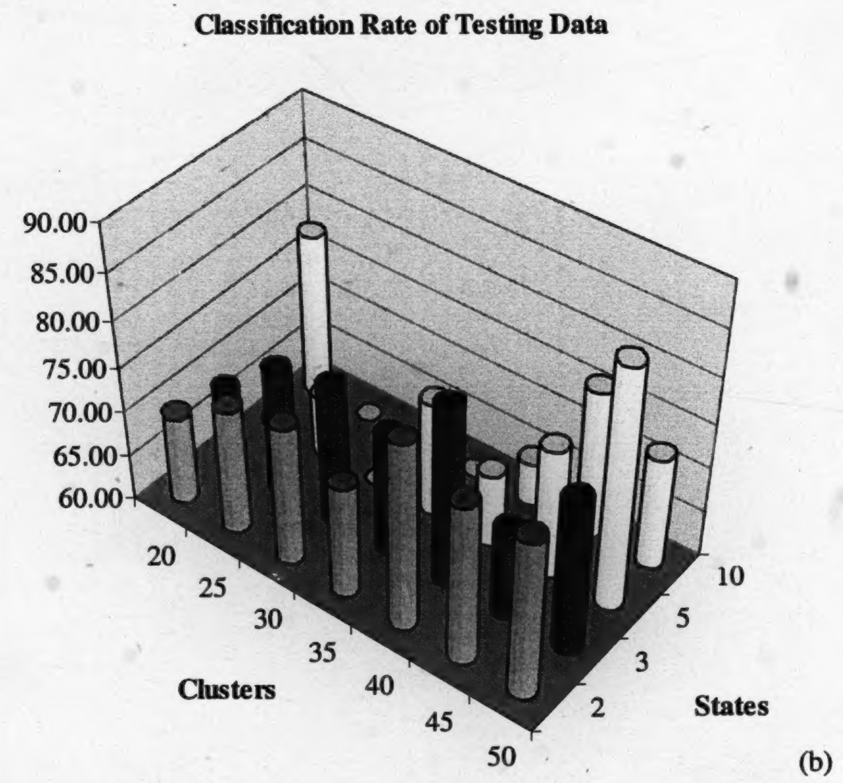


Figure 44 – Classification Rate According to Number of States and the Number of Prototypes for Training (a) and Testing (b) Data with 15 Gestures Recognized

Again, one can see that the solution space is multimodal.

In order to gain a better understanding of what is causing classification errors, we can look at the confusion matrixes. First examine the confusion matrix for the testing set with 8 gestures. We should see 80 on the diagonal since there are a total of 80 of each gesture being recognized. The following problems exist:

- First the capture gestures are sometimes confused with each other. These gestures both cover all sensors.

- Second, confusion occurs between the "Stay Right" and "Go Left" gestures. These gestures are primarily at the same location on the robot.
- Second the "Go left" the "Turn Counter-Clockwise". These gestures begin the same way, but "Go Left" is shorter.
- There is also some confusion between the turning gestures. These gestures both cover all sensors.

		Gesture Recognized							
		Capture Front Back	Capture Sides	Go Left	Go Right	Stay Left	Stay Right	Turn Counter-Clockwise	Turn Clockwise
Gesture Performed	Capture Front Back	71	5	1	0	0	0	0	3
	Capture Sides	5	63	0	0	3	6	0	3
	Go Left	0	0	78	0	0	0	2	0
	Go Right	0	0	2	77	0	0	0	1
	Stay Left	0	0	0	7	73	0	0	0
	Stay Right	0	5	1	0	0	74	0	0
	Turn Counter-Clockwise	0	0	5	0	0	0	71	4
	Turn Clockwise	0	0	2	4	0	0	1	73

Table 19 – Misclassification Matrix for 8 Gestures (Testing Data)

We will now present the confusion matrix for the classification of 15 gestures. Again, we should see 80 on the diagonal.

		Gesture Recognized														
		Capture front back	Capture sides	Go left	Go right	Stay left	Stay right	Turn clockwise	Turn counterclock	Hit back	Hit front	Hit left	Hit right	Tickle back	Tickle front left	Tickle front right
Gesture Performed	Capture front back	75	0	1	0	0	0	0	0	2	0	0	0	2	0	0
	Capture sides	0	58	0	0	0	0	2	1	0	0	2	15	1	0	1
	Go left	0	0	78	0	0	0	2	0	0	0	0	0	0	0	0
	Go right	0	0	4	74	0	0	0	1	0	0	0	0	0	1	0
	Stay left	0	0	0	5	68	0	0	0	0	0	5	0	0	2	0
	Stay right	0	0	1	0	0	29	1	0	0	0	0	18	0	0	31
	Turn counter-clockwise	0	0	7	1	0	0	68	3	1	0	0	0	0	0	0
	Turn clockwise	0	0	1	4	0	0	1	73	1	0	0	0	0	0	0
	Hit back	0	0	0	0	0	0	0	0	73	0	0	0	7	0	0
	Hit front	0	0	2	0	0	0	0	0	0	78	0	0	0	0	0
	Hit left	0	0	0	0	0	0	0	0	0	0	78	0	0	2	0
	Hit right	0	0	1	0	0	0	0	0	0	0	0	79	0	0	0
	Tickle back	0	0	0	0	0	0	0	0	10	0	0	0	70	0	0
	Tickle front left	0	0	0	2	1	0	0	0	0	1	10	0	0	66	0
	Tickle front right	0	0	2	0	0	4	0	0	0	0	0	1	0	0	73

Table 20 - Confusion Matrix for the Classification of 15 Gestures (Testing Data)

There are frequent misclassifications that occur when the same sensors are activated, when the gestures are similar, and/or when they begin the same way.

We will also look at the case when we only attempt to categorise according to the 4 major gesture categories. The problems mentioned above continue to create difficulties even when we merge the gestures into groups. Here, however, the problems are more likely to occur when the same sensors are activated rather than because of other similarities.

		Group Recognized			
		caress	direct	attack	play
Group Performed	caress	133	4	19	4
	direct	0	421	25	34
	attack	0	3	**308**	9
	play	0	9	22	209

Table 21 – Confusion Matrix for Classification of 4 Major Categories of Gestures

The category that is best identified is the attack gesture. This is not surprising since the gestures involve entering and leaving the range of the sensors repeatedly. This is the only gesture of that nature. The caress category is the worst identified category. This is likely because the gestures from this category cover all of the sensors on the robot and thus slight variations in gesticulation could cause them to be confused with another gesture.

8.3 Chapter Summary

We have presented the results of the gesture recognition system. We have seen that the solution space is multimodal and that changing parameters can produce both positive and negative effects. Finally, we have seen that misclassification tends to occur for one of two reasons, either the postures are not easily distinguished when the same sensors are activated or the gestures follow a similar temporal path for part of the gesticulation.

Chapter 9: Conclusions and Future Work

Let us now examine the future path of this research and present concluding comments on the research and results.

9.1 Future Work – The Robot's Response

We will now propose a possible model for how the robot could respond and interact with a human based on the gestures of the human and internal motivations. This chapter will present some background information on current robotic systems that model human robot interactions with emotional models. We will explore possible robot emotional states and their corresponding behaviour. We present some proposed motivational drives for the robot and then explore how a limited memory system could be implemented. Finally, we will present a suggested state transition model that account for how the robot travels from one emotional state to another.

9.1.1 Background

We want to explore an example of how Hèbert could respond to the human gestures and construct a behavioural model based on emotions and motivations. Sony's AIBO and Kismet [5, 6, 39] both use the concept of emotions and motivations in the interactive robot behaviour engines. A combination of internal robot motivations, environmental stimulation, including human interaction, and time alter the emotional state of the robot. The robots behave according to their current emotional state.

Kismet is able to display its emotional state and behaviour by altering its facial features and audio output. AIBO changes the status of a number of LEDs, its motor response, and audio output.

9.1.2 Emotional states

Kismet has 7 different emotional states and AIBO has 6. For Hèbert, we propose a model with 8 different emotional states. We could represent its current emotional state by changing the status of its LEDs, altering its responsiveness to commands, and changing both the speed a type of movement. Table 14 shows emotional states and their associated behaviours.

Emotion	Responsiveness to Commands	LEDs	Motion Speed	Inherent Movement
Happy	Obedient	Medium blinking	Medium	Still
Excited	Mostly Obedient (Obeys 4 out of 5 commands)	Fast blinking	Quick	Twitches
Curious	Must capture its attention	Mostly on with quick blinks off	Quick	Explores constantly
Bored	Obedient	Slow blinking	Slow	Still
Sad	Less responsive (Obeys every 2nd command)	Slow blinking with more time off than on	Slow	Still
Afraid	Unresponsive	Wide eyes, always on	Quickly	Runs away from hands (darkness).
Annoyed	Less responsive (Obeys every 3rd command)	Slowly alternating	Medium	Moves away from hands (darkness) when disobeying

Emotion	Responsiveness to Commands	LEDs	Motion Speed	Inherent Movement
Angry	Unresponsive	Quickly alternating	Quickly	Charges hands (darkness)

Table 22 – Emotional States and Associated Behaviours

9.1.3 Motivations

In this model, Hèbert would have two primary motivations. The first is for positive interactions and then second is exploration. When it experiences positive interactions, its emotional state tends towards happiness. Conversely, when it experiences negative interactions, its emotional state would tend towards anger. When it experiences no interactions, it would tend towards boredom.

Its perception of whether an interaction is positive or negative also depends on its current emotional state. For example, if it was annoyed and you commanded it to do something, it would perceive that negatively, but if it was bored and you commanded it to do something it would perceive that positively.

The second motivating drive for Hèbert is to explore and move around. If the robot has not moved in a set period of time, it will become curious and begin to explore. After it has satisfied its need to explore, it will become happy and stop exploring.

9.1.4 Memory

The robot would change emotional states based on the number of times a given category of gesture was perceived while in that state and the strength of certain the motivating drives. The state transition model inherently has some memory built into it because you can only reach certain states through certain paths.

We will model two other types of memory: very short-term memory and short-term memory. We will use very short-term memory to track the number of time each category of gestures has occurred. These counters are reset each time the robot changes from one emotional state to another.

Short-term memory is what tracks the strength of the motivating drives. Time spent interacting increases the score for tracking the interaction drive and time spent without interacting decreases it. The lower the score, the more the Hèbert desires interaction. Time spent moving decreases the urge to explore drive and time elapsed staying still increases it.

9.1.5 State Transitions

We have already presented the emotional states of the robot, now we will propose how Hèbert could move from one state to another. Table 23 is a state transition table. The top entry in each row represents the threshold level for each counter (number of occurrences of a gesture or current level of the drive) and the bottom row shows the next state. For example, in the "Happy" state, if the robot detects a caress gesture more than 10 times it will move to the "Excited" state. The table is only an example of what the state transition table could be.

Recall that the counters for the "Caress", "Attack", "Play", and "Direct" gestures are reset every time Hèbert makes a state transition. The interaction drive counts up or down every second, depending on whether or not the robot detects an interaction or not. The exploration drive counts up every second that the robot does not move to a new location, and counts down every time he does.

State	Caress	Attack	Play	Direct	Interaction Drive	Exploration Drive
Happy	> 10 Excited	> 5 Sad	> 10 Excited	> 20 Annoyed	< 10 Bored	< 5 Curious
Excited	> 10 Happy	> 5 Sad	> 30 Annoyed	> 20 Happy	< 20 Happy	< 1 Curious
Curious	> 1 Happy	Don't Care	Don't Care	Don't Care	< -30 Bored	> 30 Happy
Bored	> 5 Happy	> 2 Afraid	> 2 Happy	> 10 Happy	< 5 Sad	< 5 Sad
Sad	> 15 Happy	> 2 Annoyed	> 6 Happy	Don't Care	Don't Care	< 1 Curious
Afraid	> 1 Afraid	> 10 Angry	> 5 Afraid	Don't Care	< 5 Bored	Don't Care
Annoyed	> 10 Sad	> 2 Angry	Don't Care	Don't Care	< 10 Bored	< 5 Curious
Angry	> 10 Sad	Don't Care	Don't Care	Don't Care	< 15 Annoyed	Don't Care

Table 23 – Emotional State Transitions

Changing the entries of the table would result in changing the "personality" of the robot. For example if it took less "Attack" gestures to move the robot towards the "Angry" state the robot would seem quick-tempered. If it took more, the robot would seem more passive.

The entries in the state transition table could be learned through reinforcement learning, genetic algorithms, fuzzy expert systems, or some other type of machine learning algorithm. [5, 6, 39] both use learning to shape the personality of the robot.

9.2 Conclusion

Human Robot Interaction is an important field of research for a number of reasons. It enables more effective service robots, provides insight into human interactions, and makes toy robots more entertaining. Progress towards better interfaces in computers has made them more accessible, commonplace, and helpful for society. Robots will become more accessible, helpful, and entertaining in society as HRI develops.

People communicate naturally with gestures. It makes sense to exploit this natural tendency as a method for interacting with robots. HRI will continue to improve with gesture recognition algorithms that are more accurate and more comprehensive.

We attempted to recognize 15 different gestures from 4 major categories: "Attack", "Caress", "Play", and "Direct". Ultimately, we would like to be able to clearly recognize gestures using simple and cheap sensors, such a IR sensors.

We implemented the gesture recognition algorithm on a Khepera robot produced by K-team. The robot has only 8 IR sensors. Using these sensors, we were able to extract gesture features with very little pre-processing compared to systems that use cameras. These sensors provided the advantage of not restricting the hands with data gloves. There is a limit to the simplicity of these sensors; certainly, results that are more accurate would be possible with sensors that had more range. Some of the gestures such as, hitting, tickling, and embracing are well suited to close range sensors. However, gestures that involve direction commands would feel more natural if they did not have to be performed within 4 cm of the robot.

We used pre-processing to extract relevant information from the data. This served two purposes: firstly, to try to increase classification rates and secondly, to reduce the computational requirements of future steps. The pre-processing stage categorized each individual sensor reading as a hand being “Absent”, “Near”, “Very near”, or “Extremely near”. These boundaries of these categories were determined based on both the distribution of the data and the relationship of the sensor values to hand proximity.

Clustering provides a feasible way to extract human hand postures, which is an important step towards modelling the spatial component of gestures. We used clustering to find posture groups from the pre-processed inputs. We then mapped inputs to the resulting prototypes of the posture groups. This allowed us to pass on the label of the group to a temporal model

We compared various clustering techniques before determining to implement FCM clustering in this stage of the recognition system. The FCM method was chosen because it is well suited to the data structure, because it is robust, and because it readily generates the prototypes we require in the classifier system.

After reviewing generic FSM and discrete Markov processes, we chose to implement HMM because their many statistical parameters offer tolerance for variation in the observation sequence while modelling a gesture and because they offered the structure of model that we required. HMMs are a proven method for modelling the temporal transitions in gestures. HMMs with discrete valued observation sequences were selected over ones with multivariate observation sequences because they are computationally simpler.

One HMM was built to model each gesture. We experimented by generating models with a number of different states and posture groupings. In the recognition system, the observation sequence of a gesture was compared against each model; we identified the gesture as the one whose model most probably matched the observation sequence.

We tested the accuracy of the recognition system by attempting to recognize both training and testing data sets. As a result, we obtained the classification rates shown in Table 24. Classification was considered successful when an observation sequence was recognized as the correct gesture. The classification rates reflect the values obtained when attempting to differentiate between:

- the two major categories of gestures, "Caress" and "Direct",
- the four major categories of gestures, "Attack", "Play", "Caress", "Direct",
- 8 gestures belonging to the "Caress" and "Direct" categories, and
- 15 gestures belonging to all major categories of gestures.

	Training	Testing
2 Categories	98.1%	95.9%
4 Categories	90.3%	89.3%
8 Gestures	93.3%	90.6%
15 Gestures	88.7%	86.7%

Table 24 – Summary of Classification Rates

Lastly, we proposed introduced the concept of emotional states and behaviours to model human robot interaction. We have briefly shown some of the current research and commercial implementation in this area. We presented possible emotional states and corresponding behaviours for Hèbert. Further to that, we presented a possible state transition table that essentially models the "personality" of the robot in terms of human robot interaction. Finally we proposed how there is room for machine learning in modeling the state transition table. Certainly, this area commands further research.

There is room for improvement in the recognitions system. Sensors with a longer range would allow movement that is even more natural from the human, as well as include information that is currently lost outside of the 4 cm range. Improved clustering would provide a better representation of the postures. Improvements could come from slight changes in the parameters of the current clustering algorithm, such as smaller values for the stopping threshold and the exponent of the partition matrix. It would also be beneficial to produce more accurate temporal models. More methodical initialization of the parameters and initial values of the HMMs would likely increase the overall performance of the recognition system.

Ultimately, we see that gesture recognition is an important part of HMI and that improved HMI is required in order to have machines, including robots, become more useful and usable in society. The use of the Khepera robot and the HMM algorithm should help to improve IR sensor gesture-based interactions.

Bibliography

1. A. Agah. "Interactions of an Anthropomorphic Simulated Human with a Simulated Service Robot," Simulation vol. 72, no. 1, pp. 12-19. January 1999.

2. R. C. Arkin. *Behaviour Based Robotics*. MIT Press. Cambridge. 1998.

3. J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press. New York. 1981.

4. P. Booth. *An Introduction to Human-Computer Interaction*. Lawrence Erlbaum Associates Ltd. East Sussex, U.K. 1989.

5. C. Breazeal. "Robot in Society: Friend or Appliance?" *Agents99 workshop on emotion-based agent architectures*, pp. 18-26. Seattle, WA. 1999.

6. C. Breazeal and B. Scassellati., "Infant-like Social Interactions between a Robot and a Human Caretaker," *Adaptive Behavior*. vol. 8, issue 1. pp. 49-74. Winter 2000.

7. R. A. Brooks. "New Approaches to Robotics," *Science*. vol. 253. pp. 1227-1232. September 1991.

8. Brooks, R.A., "The Cog Project," *Journal of the Robotics Society of Japan, Special Issue on Humanoid*, Vol. 15, No. 7. Toshihiro Matsui, editor. October 1997.

9. *Canadian Oxford Dictionary*. Oxford University Press. 1998.

10. H. I. Choi, P. K. Rhee. "Head Gesture Recognition using HMMs," *Expert Systems with Applications*, vol. 17, no. 3, pp. 213-221. 1999.

11. K. J. Cios, W. Pedrycz, R. W. Swiniarski. *Data Mining Methods for Knowledge and Discovery*. Kluwer Academic. Boston. 1998.

12. A. W. Drake. *Fundamentals of Applied Probability Theory*, Chapter 5. McGraw-Hill. New York, NY. 1967.

13. R.O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis*, Chapter 6. Wiley-Interscience. New York. 1973.

14. P. Emeagwali. http://emeagwali.com/essays/technology/computing/history-of-computing-abacus-logarithm-eniac-supercomputers.html

15. S. Haykin. *Neural Networks a Comprehensive Foundation*, Chapter 10. 1994.

16. E. J. Holden, G. Roy, R. Owens. "Adaptive Classification of Hand Movement," *Proceedings of IEEE International Conference on Neural Networks*, vol. 3, pp. 1373-1378. 1995.

17. Honda. http://world.honda.com/news/2002/c021205.html. December 2002.

18. A. J. Howell, H. Buxton. "Active Vision Techniques for Visually Mediated Interaction," *Image and Vision Computing*, vol. 20, no. 12, pp. 861-871. 2002.

19. C. L. Huang, M. S. Wu, S. H. Jeng. "Gesture recognition using the multi-PDM method and hidden Markov model," *Image and Vision Computing*, vol. 18, no. 11, pp. 865-879. 2000.

20. S. Iba, et al. "An Architecture for Gesture-Based Control of Mobile Robots," *Proceedings of the 1999 IEEE/RJS International Conference of Intelligent Robots and Systems*, pp. 851-857. 1999.

21. Y. Iwai, T. Hata, M. Yachida, "Gesture Recognition based on Subspace Method and Hidden Markov Model," *Proceedings of International Conference of Intelligent Robots and Systems (IROS'97)*, Vol. 2, pp. 960-966. Grenoble, France. September 1997.

22. B. H. Juang, et al. "Maximum Likelihood Estimation for Mixture Observations of Markov Chains," *IEEE Transactions on Information Theory*, vol. IT-32, no. 2, pp. 307-309. March 1986.

23. A. Junghanns. "Using Search in Knowledge-Based Engineering," *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI'00)*. Berlin, Germany. August 2000.

24. K-team. *Khepera User Manual*. www.k-team.com

25. O. Khabit, et al. "Robots in Human Environments: Basic Autonomous Capabilities." *International Journal of Robotics Research,* vol.18, no.7 p.684-96. 1999.

26. V. Klingspor. "Human-Robot-Communication and Machine Learning," *Applied Artificial Intelligence Journal.* vol. 11, pp. 719-746. 1997.

27. Rung-Huei Liang, Ming Ouhyoung, "A Real-time Continuous Gesture Interface for Taiwanese Sign Language," submitted to UIST97. March 1997.

28. C. C. Lien, C. L. Huang. "Model-based articulated hand motion tracking for gesture recognition," *Image and Vision Computing*, vol. 16, no. 2, pp. 121-134. 1998.

29. L. A. Liporace. "Maximum Likelihood Estimation for Multivariate Observation of Markov Sources," *IEEE Transactions on Information Theory*, vol. IT-28. no. 5, pp. 729-734. September 1982.

30. B. Mirkin. *Mathematical Classification and Clustering*, Chapter 3. 1996.

31. C. Möller. http://users.design.ucla.edu/projects/arc/cm/cm/staticE/page35.html. 1994.

32. H. Moravec. *Mind Children – The future of Robot and Human Intelligence.* Harvard University Press Cambridge, Massachusetts. 1988.

33. R.G. O'Hagan, A. Zelinsky, S. Rougeaux. "Visual Gesture Interfaces to Virtual Environments," *Interacting with Computers, Special Issue.* 2001.

34. W. Pedrycz, A. V. Vasilakos. "Linguistic Models and Linguistic Modeling," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, Vol. 29, No. 6. December 1999.

35. A. Psarrou, S. Gong, M. Walter. "Recognition of human gestures and behaviour based on motion trajectories," *Image and Vision Computing*, vol. 20, no. 5-6, pp. 349-358. 2002.

36. L.R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceeding of the IEEE*, Volume 77, pp. 257-285. February 1989.

37. B. Scassellati. "Investigating Models of Social Development using a Humanoid Robot," *AAAI Fall Symposium Robots and Biology: Developing Connections*. Orlando, Fl. 1998.

38. B. Selic, G. Gullekson, P. T. Ward. *Real-time object-oriented modeling*. Wiley & Sons. New York. 1994.

39. Sony. AIBO. http://www.world.sony.com/Electronics/aibo/index.html

40. T. Starner and A. Pentland. "Visual Recognition of American Sign Language Using Hidden Markov Model," *Proceedings International Workshop on Automatic Face and Gesture Recognition*, pp. 189-194. June 1995.

41. K. Suzuki, et al. "Intelligent Agent System for Human-Robot Interaction through Artificial Emotion," *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics*, vol.2, pp.1055-60. 1998.

42. S. Tamura, S. Kawasaki. "Recognition of sign language motion images," Pattern Recognition, vol. 21, no.4, pp. 343-353. 1988.

43. J. Triesch, C. von der Malsburg. "Classification of hand postures against complex backgrounds using elastic graph matching," *Image and Vision Computing*, vol. 20 no.13-14, pp. 937-843. 2002.

44. P. Wallich. "The Ghosts of Computers Past," *IEEE Spectrum*, vol.39, no.11, pp.33-43. November 2002.

45. W. G. Walter. *The Living Brain*. Norton. New York. 1953.

46. Webots Simulator. www.cyberbotics.com

47. D. M. Wilkes, et al. "Designing for Human-Robot Symbiosis," *Industrial Robot*, vol.26, no.1, pp.49-58. 1999.

48. H.S. Yoon, J. Soh, B.W. Min, H.S. Yang. "Recognition of Alphabetical Hand Gestures Using Hidden Markov Model," *IEICE Transactions Fundamentals*, vol.E82, no. 7, pp. 1358-1366. July 1999.

49. H.S. Yoon, J. Soh, Y. J. Bae, H.S. Yang. "Hand Gesture Recognition Using Combined Features of Location, Angle, and Velocity," *Pattern Recognition*, vol. 34, no. 7, pp. 1491-1501. 2001.